Découvrir Python sur la TI-Nspire™ CX II-T

Auteur : Jean-Louis Balas

Texas Instruments

Sommaire

Unité 1 : Débuter la programmation en Python

Compétence 1 : Calculer avec Python

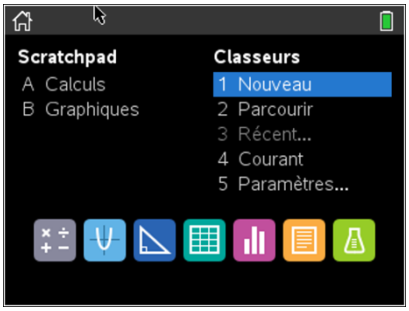
Dans cette première leçon de l'unité 1, vous allez découvrir l'application TI-Python en utilisant les fonctions mathématiques les plus courantes implémentées dans la calculatrice TI-Nspire™ CX II.

Objectifs :

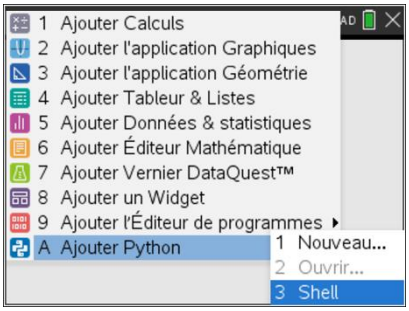
- Utiliser le module TI-Python.
- Découvrir les fonctions mathématiques en Python.
- Distinguer l'éditeur de programmes et la console (Shell).
- Utiliser une instruction de programmation dans la console.

TI-Nspire CX II.

A partir de l'écran d'accueil de la calculatrice (ou du logiciel), créer un nouveau document.

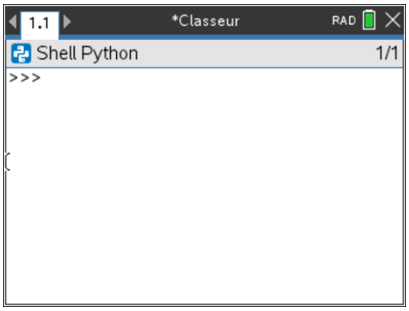


Choisir le menu **A** puis **3 Shell** afin d'accéder à la console.



Vous pouvez directement entrer le nombre **3,** ce qui est plus rapide que l'utilisation du « trackpad » afin d'afficher le « prompt » de la console.

La ligne active et le nombre total de lignes sont respectivement indiqués par le numérateur et le dénominateur de la fraction située en face de : Shell Python.



Conseil à l'enseignant : L'utilisation de l'application TI-Python nécessite la mise à jour de l'OS de la calculatrice vers la version 5.2 ou supérieure. Les programmes développés à partir d'un éditeur Python peuvent également être transférés directement lorsque la calculatrice est connectée à l'ordinateur, par l'intermédiaire du câble USB.

Unité 1 : Débuter la programmation en Python

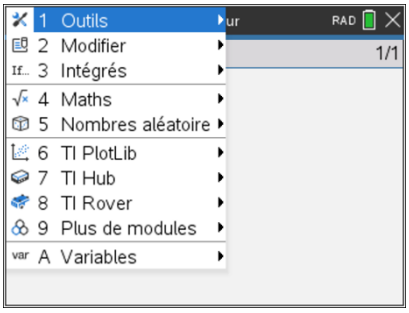
Compétence 1 : Calculer avec Python

Dans cette première leçon de l'unité 1, vous allez découvrir l'application TI-Python en utilisant les fonctions mathématiques les plus courantes implémentées dans la calculatrice TI-Nspire™ CX II.

Objectifs :

- Utiliser le module TI-Python.
- Découvrir les fonctions mathématiques en Python.
- Distinguer l'éditeur de programmes et la console (Shell).
- Utiliser une instruction de programmation dans la console.

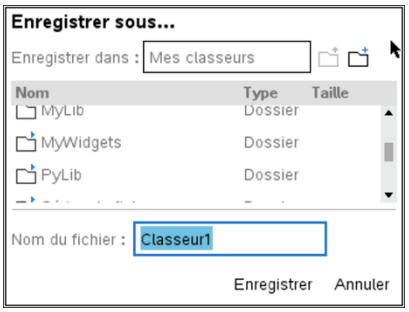
La touche [menu] de la calculatrice permet d'accéder à l'ensemble des fonctionnalités (rubriques 1 à 3) du langage Python.
Les rubriques 4 à 9 donnent accès aux bibliothèques intégrées (Maths ; Nombres aléatoires, TI PlotLib etc.).

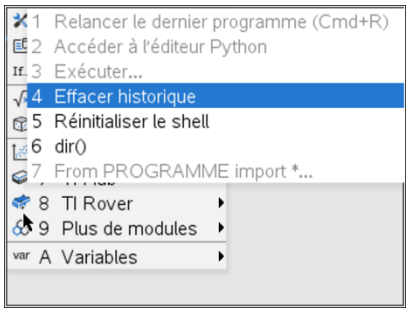


La touche [var] est particulièrement importante, car elle permettra d'accéder facilement à toutes les variables créées, soit en mode console, soit lors de l'écriture d'un script Python.

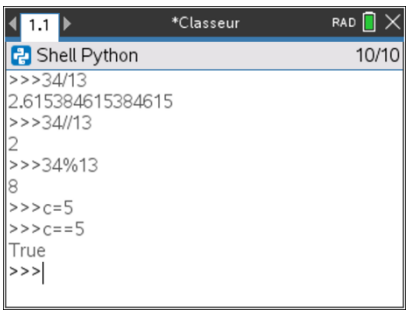
Conseil à l'enseignant : L'utilisation du langage Python s'effectue généralement à partir d'un script que l'on exécute dans la console. Cependant, dans la console, il est possible de :

- Faire des calculs, définir des variables afin de les intégrer dans des calculs.
- Écrire et exécuter un programme.
- Exécuter un programme saisi dans l'éditeur et demander les valeurs prises par les variables de ce programme.
- En appuyant sur la touche [doc] puis **1 Fichier** puis **5 Enregistrer sous** un script Python peut être intégré au répertoire PyLib afin d'être utilisé ultérieurement comme une bibliothèque.
- Le script sera ensuite intégré à partir du menu **1 Outils** puis **7 From PROGRAMME import*...**, lequel sera actif dès lors que le dossier **PyLib** ne sera pas vide.





Dans un premier temps nous allons utiliser la console également appelée « Shell »



Quelques commandes de base.

Les variables sont généralement nommées par des lettres minuscules.
$c \leftarrow 5$ va s'écrire en Python $c = 5$ et s'obtient sur la calculatrice en tapant : (c [=] 5). **Cette instruction signifie que la valeur 5 est affectée à la variable nommée c.**
Pour tester la valeur de la variable c : on écrira $c == 5$ ou bien $c >= 5$ etc.

Unité 1 : Débuter la programmation en Python

Compétence 1 : Calculer avec Python

Dans cette première leçon de l'unité 1, vous allez découvrir l'application TI-Python en utilisant les fonctions mathématiques les plus courantes implémentées dans la calculatrice TI-Nspire™ CX II.

Objectifs :

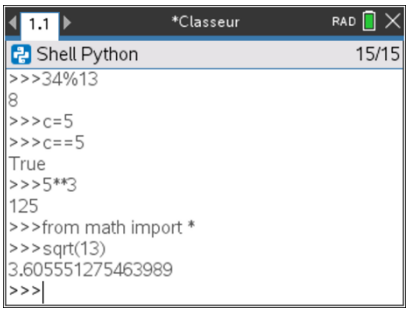
- Utiliser le module TI-Python.
- Découvrir les fonctions mathématiques en Python.
- Distinguer l'éditeur de programmes et la console (Shell).
- Utiliser une instruction de programmation dans la console.

Les calculs classiques :

- Le reste de la division de a par b s'écrit **a%b**.
- Le quotient euclidien de a par b s'écrit **a //b**.
- x à la puissance n s'écrit **x**n**. On peut aussi écrire **pow(x,n)**.

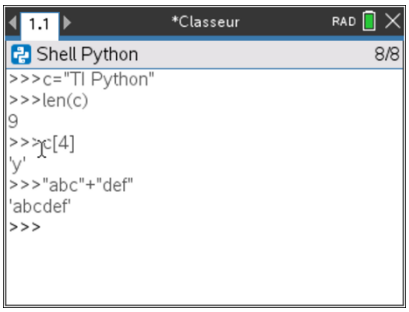
Remarque : Le chargement du module (bibliothèque) « math import » est nécessaire pour effectuer des calculs sur les racines carrées et sur les fractions.
Pour incorporer ce module, appuyer sur la touche [menu] et choisir 4 Maths puis enfin le menu 1 from math import*

- La racine carrée de x $(x \geq 0)$ s'écrit **sqrt(x)**.
- Le nombre π s'écrit **pi**.



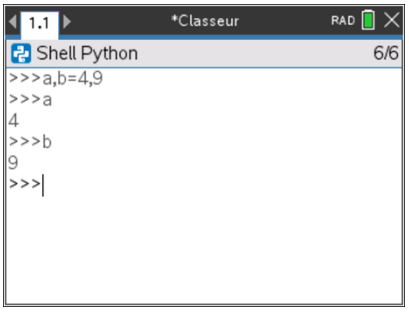
Les commandes relatives aux chaînes de caractères.
Les chaînes de caractères se définissent à l'aide de guillemets doubles ou simples : « TI-Python » ou bien 'TI-Python'

- Obtenir la longueur d'une chaîne de caractères **len(c)** (Menu **Fns…** puis **List**).
- c[k] renvoie le k+1ème élément de la chaîne c.
- Pour concaténer deux chaînes de caractères, simplement les additionner.



Remarque : Pour effacer une console des évènements précédents, appuyer sur [menu] et choisir le menu **1 Outils** puis **4 : Effacer l'historique**.
Les variables et leurs contenus seront conservés.
Si vous ne souhaitez pas conserver les variables définies, choisir l'option **5 Réinitialiser le Shell**.

Un peu d'astuce :
Lors de l'affectation de plusieurs variables, il est possible de le faire en une seule fois comme le montre l'écran ci-contre.



Unité 1 : Débuter la programmation en Python

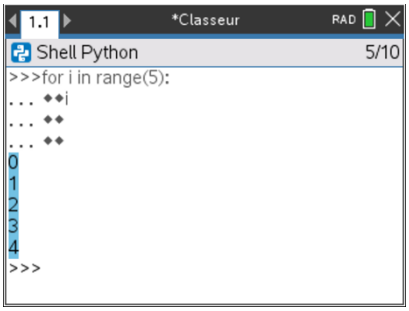
Compétence 1 : Calculer avec Python

Dans cette première leçon de l'unité 1, vous allez découvrir l'application TI-Python en utilisant les fonctions mathématiques les plus courantes implémentées dans la calculatrice TI-Nspire™ CX II.

Objectifs :

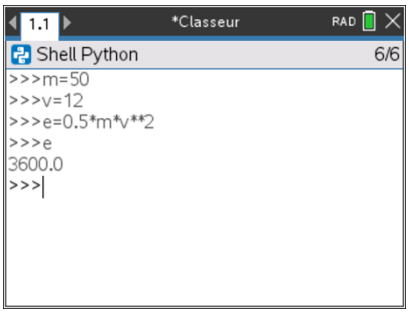
- Utiliser le module TI-Python.
- Découvrir les fonctions mathématiques en Python.
- Distinguer l'éditeur de programmes et la console (Shell).
- Utiliser une instruction de programmation dans la console.

Utiliser une instruction de programmation dans le Shell.
Le langage Python possède la richesse de pouvoir observer, indépendamment d'un script, une fonctionnalité particulière.
Ainsi sur l'écran de droite, on peut analyser le fonctionnement d'une boucle **for** à laquelle on accède en appuyant sur menu puis **3 : Intégrés** et enfin **2 : Contrôle** en choisissant dans le menu l'option **4 : for index in range(size) :**
Nous reviendrons sur les boucles dans une leçon ultérieure.

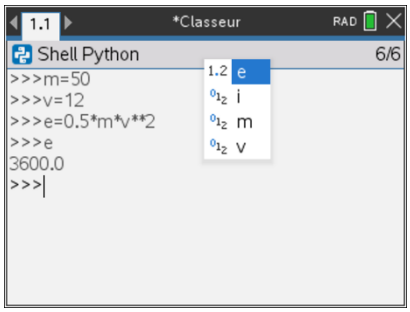


Conseil à l'enseignant : l'utilisation directe du clavier de la calculatrice permet également d'écrire les commandes en Python. La coloration syntaxique s'active automatiquement lorsque les instructions sont écrites sans erreur.
Le logiciel TI-Nspire™ CX pour ordinateur permet également l'utilisation directe du clavier de l'ordinateur.

Appliquons nos connaissances.
L'énergie cinétique d'un solide en mouvement est donnée par la relation $E_c = \frac{1}{2}mv^2$
m est la masse du solide en kg.
v est sa vitesse en m/s.
En utilisant la console, quelle est la valeur de la variable *energie* (e) si la *masse* est de 50 kg et la *vitesse* de 12 m/s ?



Conseil à l'enseignant : Un programme informatique contient des instructions qui utilisent des variables. Une variable est une « case » qui permet de conserver des données du programme (nombre, valeur entrée par l'utilisateur, chaîne de caractères …) en les stockant dans la mémoire de l'ordinateur. L'affectation d'une valeur dans une variable se fait à l'aide de la touche = qui recopie dans le Shell le signe **=** .
La calculatrice TI-Nspire CX II permet d'accéder à l'ensemble des variables utilisées dans un script, mais aussi dans le mode console. Pour cela, appuyer sur menu puis choisir **A Variables** et enfin **2 Variables : Tout** .



Unité 1 : Débuter la programmation en Python

Compétence 2 : Les types de données en Python

Dans cette deuxième leçon de l'unité 1, vous allez découvrir comment utiliser le type des données en Python.

Objectifs :

- Connaitre les différents types de données en langage Python
- Mettre en forme le format d'une donnée numérique

Connaitre le type des grandeurs utilisé.

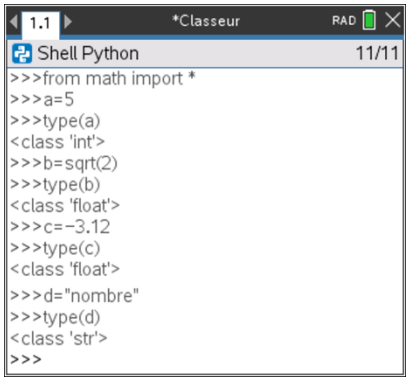
Lorsque vous utilisez un script en langage Python, il peut s'avérer nécessaire de connaitre le type de variable utilisé, ou bien de modifier ces variables en vue d'une utilisation ultérieure. Par exemple si la grandeur renvoyée par un script Python renvoie une grandeur correspondante à une mesure en sciences physiques, il n'est pas nécessairement opportun de conserver un résultat à 6 décimales.

Vous allez créer un script dans l'application **Python** permettant de distinguer les types des grandeurs utilisées.

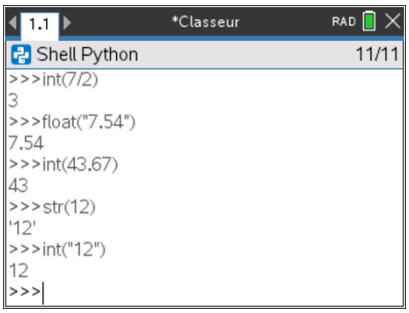
- Une chaîne de caractères.
- Un nombre réel.
- Un nombre irrationnel, $\sqrt{2}$ par exemple.

Vous afficherez le nombre ainsi que son type à l'aide de l'instruction **type**.

- Créer un nouveau document Python comportant uniquement la console.
- Importer le module **maths** (menu puis **4 Maths**) et enfin **1 from maths import***.
- L'ensemble des commandes utilisées peut être directement entré au clavier. La commande type qui permet de connaître la nature d'une variable est tapée à la main.



Astuce : L'utilisation des touches de direction ▲ puis **enter** permet de recopier une ligne lorsque l'utilisateur travaille dans la console.



Remarques :

- L'opérateur **int()** extrait lorsque c'est possible, un entier d'une chaîne de caractères et renvoie la partie entière d'un nombre comprise entre $\pm$ 2 147 483 648 (codage sur 32 bits, soit 4 octets)
- L'opérateur **str()** transforme un nombre en une chaîne de caractères.
- L'opérateur **float()** extrait lorsque c'est possible, un flottant d'une chaîne de caractères.

Unité 1 : Débuter la programmation en Python

Compétence 2 : Les types de données en Python

Dans cette deuxième leçon de l'unité 1, vous allez découvrir comment utiliser le type des données en Python.

Objectifs :

- Connaitre les différents types de données en langage Python.
- Mettre en forme le format d'une donnée numérique.

Une autre astuce :

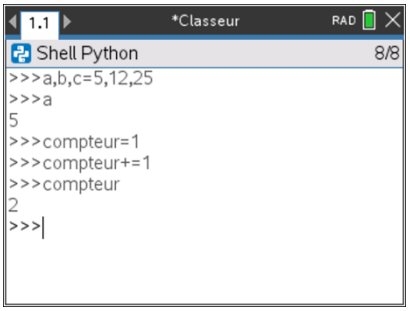
Pour incrémenter une variable dans un compteur, on dispose de deux possibilités :

a) Écrire par exemple :

```
compteur = 0
compteur = compteur +1
demander l'affichage de la variable compteur
```

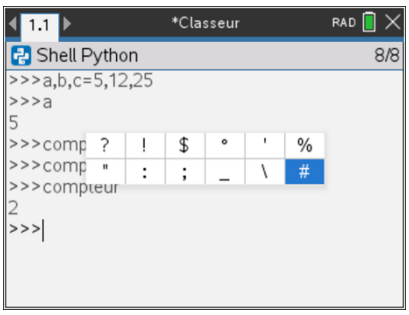
b) Ou bien

```
compteur = 0
compteur+=1
compteur
```



Conseils à l'enseignant : Utiliser les fonctions de « copier-coller » **Ctrl C** et **Ctrl V**
Lors de la rédaction, la touche del permet d'effacer un caractère entré par erreur.

Vous avez la possibilité de commenter vos scripts en ajoutant devant le commentaire, un (# commentaire). Le fait de mettre un # indique que la ligne ne sera pas interprétée. Pour l'obtenir, appuyer sur la touche ?!



Unité 1 : Débuter la programmation en Python

Compétence 2 : Les types de données en Python

Dans cette deuxième leçon de l'unité 1, vous allez découvrir comment utiliser le type des données en Python.

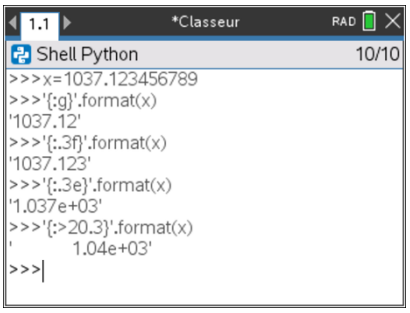
Objectifs :

- Connaitre les différents types de données en langage Python.
- Mettre en forme le format d'une donnée numérique.

Pour aller plus loin :

Le format des nombres : La méthode **format** de l'objet string est un outil très puissant permettant de créer des chaînes de caractères en remplaçant certains champs (entre accolades) par des valeurs (passées en argument de la fonction format) après conversion de celles-ci. On peut préciser à l'intérieur de chaque accolade un code de conversion, ainsi que le gabarit d'affichage. Donnons quelques exemples.

```
>>> x=1037.123456789
>>> '{ :g}'.format(x) # choisit le format le plus approprié '1.04e+03'
>>> '{ :.3f}'.format(x) # fixe le nombre de décimales
'1037.123'
>>> '{ :.3e}'.format(x) # notation scientifique
'1.037e+03'
>>> '{0 :20.3f}'.format(x) # précise la longueur de la chaîne
'  1037.123'
>>> '{0 :>20.3f}'.format(x) # justifié à droite
'  1037.123'
>>> '{0 :<20.3f}'.format(x) # justifié à gauche
'1037.123  '
>>> '{0 :^20.3f}'.format(x) # centré
'  1037.123  '
>>> '{0 :+.3f} ; {1 :+.3f}'.format(x, -x) # affiche toujours le signe '+1037.123 ; -1037.123'
>>> '{0 : .3f} ; {1 : .3f}'.format(x, -x) # affiche un espace si x>0
```



Unité 1 : Débuter la programmation en Python

Compétence 3 : Les fonctions en Python

Dans cette troisième leçon de l'unité 1, vous allez utiliser l'éditeur de programme (script) afin de créer des fonctions, puis exécuter celui-ci afin d'observer les résultats dans la console

Objectifs :

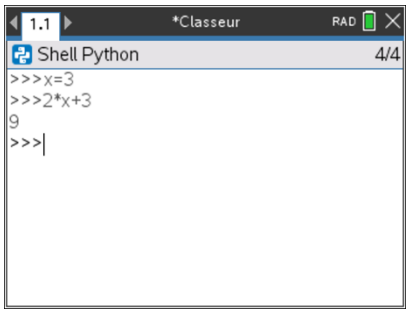
- Découvrir la notion de fonction en Python.
- Créer une fonction.

Vers la notion de fonction en Python.

Mettre en œuvre l'algorithme suivant :

$$x \leftarrow 3$$
$$y \leftarrow 2 \times x + 3$$

Dans une console, ceci se réalise très simplement.



Mais si l'on souhaite répéter ce type de calcul pour une autre valeur de x il faut écrire de nouveau l'ensemble de la séquence. Ce qui, sur un exemple moins trivial, peut s'avérer vite fastidieux.
On est donc conduit à créer une fonction qui nous permettra de dupliquer aisément le traitement de l'algorithme.

En algorithmique, une fonction peut être considérée comme une séquence d'instructions, réalisant une certaine tâche, en utilisant un ou plusieurs **arguments** (ou aucun dans certains cas).
Cette fonction reçoit un nom.

- La programmation d'une fonction commence toujours par def suivi du nom de la fonction, suivi des arguments de celle-ci. Cette ligne se termine par le symbole **:**
- Les deux points indiquent le début du bloc d'instructions définissant la fonction : toutes ces instructions sont **indentées**, c'est-à-dire décalées vers la droite par rapport à la première ligne. On ajoute en tête de chaque ligne, le même nombre d'espaces.

La fonction renvoie un seul résultat par l'intermédiaire de la commande **return.** Le résultat peut être constitué d'une liste de résultats, une chaîne de caractères etc.

```
def nom_fonction(liste des arguments) :
    bloc d'instructions
    return (résultats)
```

L'indentation, obtenue avec la touche de tabulation ou avec des espaces, est **primordiale** : tout ce qui est indenté après le def() sera exécuté comme un bloc. Il ne faut pas que l'indentation varie (nombre d'espaces, passer de la tabulation à des espaces. . .) en cours de bloc.

Conseil à l'enseignant : Une fonction permet de découper le problème étudié en sous-problèmes et d'éviter ainsi la répétition d'instructions. Une fois définie, elle peut être « appelée » tout au long de l'exécution du programme autant de fois que nécessaire.
Une fonction peut n'avoir aucun argument. Elle peut également être appelée dans un autre programme : il suffit pour cela de l'insérer dans une instruction en saisissant son nom et les valeurs des arguments.

Unité 1 : Débuter la programmation en Python

Compétence 3 : Les fonctions en Python

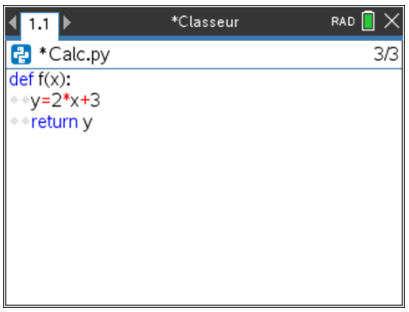
Dans cette troisième leçon de l'unité 1, vous allez utiliser l'éditeur de programme (script) afin de créer des fonctions, puis exécuter celui-ci afin d'observer les résultats dans la console

Objectifs :

- Découvrir la notion de fonction en Python.
- Créer une fonction.

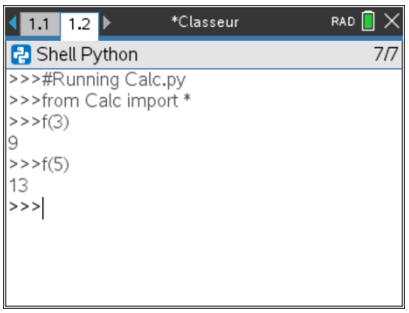
- Vous devriez obtenir l'écran ci-contre. Vous remarquerez l'indentation automatique du curseur.
- Continuez ensuite par les instructions de traitement de l'algorithme. Souvenez-vous que l'affectation d'une variable s'effectue en appuyant sur la touche **=** .
- Appuyer ensuite sur la touche menu puis **4 Intégrés**, **1 Functions** et enfin **2 : return** puis compléter l'instruction.

L'instruction peut également être tapée manuellement, la coloration syntaxique sera mise en œuvre dès reconnaissance de celle-ci.



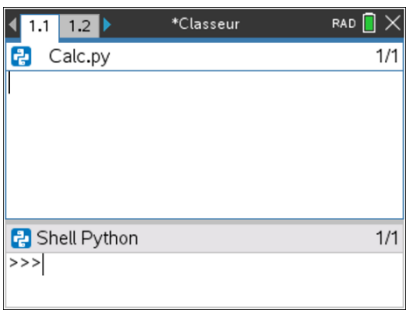
A présent vous êtes prêts pour exécuter votre script.

- Appuyer sur ctrl R.
- La console s'affiche et un message vous informe du chargement du script.
- Compléter l'invite de commande par le nom de la fonction avec les arguments prévus (un seul dans cet exemple), puis valider.
- Les touches de déplacement ▲ et **enter** permettent de rappeler une instruction.



Conseil à l'enseignant : L'édition et l'exécution d'un script Python peuvent être facilités avec la TI-Nspire en créant une seconde page (doc puis **5 : Format de page**. Choisir le format qui vous convient et, dans la seconde page, insérer une application correspondant au **Shell**.
Ainsi, vous obtiendrez une fenêtre correspondant à celle d'un éditeur complet.

Par la suite, la commande ctrl R passera directement dans la console afin d'exécuter le script, directement sous la fenêtre dans laquelle il a été édité.



Unité 1 : Débuter la programmation en Python — Application : Les différents type de données Python

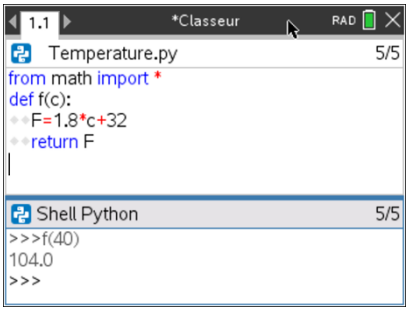
Écrire quelques scripts permettant de réinvestir les notions vues dans les leçons de l'unité 1
- Fonction en langage Python
- Création de variables numériques et chaines de caractères.

Objectifs :

- Créer un convertisseur de température.
- Créer un script permettant de développer une expression algébrique.

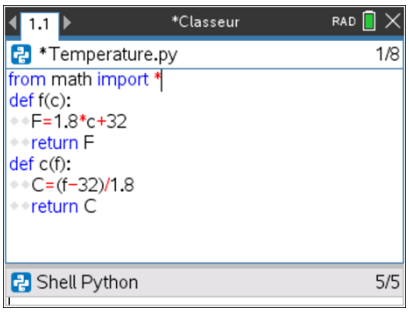
Exemple n°1 : Convertir une température.
Pour mesurer la température en France, on utilise le degré Celsius (°C). Dans les pays anglo-saxons, on utilise le degré Fahrenheit (°F).
Votre travail consiste à programmer une fonction qui réalise la conversion de température dans les deux sens : $°C \leftrightarrow °F$
Existe-t-il une température qui soit égale dans les deux unités ?
On rappelle : $t(°F) = \frac{9}{5} \times t(°C) + 32$ et en première approximation on peut utiliser $t(°F) = t(°C) \times 1.8 + 32$

- Ouvrir l'application Python et commencer un nouveau script.
- Nommer le script **Temperature** et valider en appuyant sur **enter**.
- Créer une nouvelle page (doc puis **5 Format de page**) permettant d'obtenir l'éditeur de script sur la première et la console sur la seconde.
- Importer le module **maths** (menu puis **Math…** et enfin **1 : from math import ***).
- Créer une première fonction de conversion $°C \rightarrow °F$ (menu puis **4 Intégrés** et enfin **1 Functions**).
- Utiliser la touche tab pour passer aisément d'un bloc à un autre.
- Exécuter le début (ctrl R) du script en réalisant la conversion en degré Fahrenheit d'une température de 40°C, f(40).



Conseil à l'enseignant : Lors de l'exécution d'un script (mode console (Shell), l'appui sur la touche var permet d'appeler la fonction sans arguments. Pour l'utiliser, appuyer sur **enter** puis compléter la fonction avec les arguments attendus.

- Terminer la réalisation du script en rajoutant à la suite les instructions nécessaires à la conversion $°F \rightarrow °C$.
- Remarque utile : Utiliser dans la palette Outils l'option **2 : Indent←** afin de revenir à la ligne non indentée (sinon, un message d'erreur s'affichera lors de l'exécution du script).
- En résumé, la fonction f(c) donne une température $°C \rightarrow °F$ et la fonction c(f) réalise la conversion $°F \rightarrow °C$
- Enregistrer votre script (ctrl B) : la syntaxe de celui-ci est vérifiée et si une erreur est détectée, celle-ci sera signalée. Lors de l'enregistrement, le message suivant est affiché. Temperature.py enregistré avec succès



Unité 1 : Débuter la programmation en Python

Application : Les différents type de données Python

Écrire quelques scripts permettant de réinvestir les notions vues dans les leçons de l'unité 1
- Fonction en langage Python
- Création de variables numériques et chaines de caractères.

Objectifs :
- Créer un convertisseur de température.
- Créer un script permettant de développer une expression algébrique.

Conseil à l'enseignant : Pour affiner le script, on pourra éventuellement le modifier en incitant l'élève à proposer un intervalle de variation à faire fixer par l'utilisateur, ainsi que la valeur du pas. On pourra créer une fonction « affichage », qui prend comme arguments l'intervalle et le pas.

Remarque importante : Attention lors de l'exécution d'une boucle de type for i in range(début , fin , pas) (le range s'arrête à « fin moins une valeur du pas »)

La boucle **FOR**, sera abordée lors de l'étude de l'unité 2 compétence 2.

Unité 2 : Débuter la programmation en Python

Compétence 1 : Instruction conditionnelle

Dans cette première leçon de l'unité 2, vous allez découvrir comment écrire et utiliser une instruction conditionnelle en Python.

Objectifs :

- Écrire et utiliser une instruction conditionnelle.
- Réinvestir la notion de fonction en Python.

Dans un programme, il est particulièrement fréquent d'avoir à orienter l'exécution de celui-ci en fonction de **conditions** qui affectent les différentes variables.
Une condition est un énoncé qui peut être **vrai** ou **faux**.
Par exemple : $a = b$ ou bien $a \geq b$ mais aussi n est pair sont des conditions qui sont vérifiées selon les valeurs affectées à ces variables.
Dans un programme, on peut tester une condition et selon que celle-ci est vraie ou fausse, effectuer un traitement ou un autre. On parle alors de **traitement conditionnel.**

```
If condition :
    Instruction A
else :
    Instruction B
```

Conseil à l'enseignant : En langage Python, il n'y a pas d'instruction pour indiquer la fin de l'instruction conditionnelle. C'est l'indentation qui décale vers la droite les instructions A et B.
elif est la contraction de else if
Pour tester l'égalité de deux valeurs en langage Python, on utilise le signe « == »

Exemple :
Une société de location de voitures propose à ses clients le contrat suivant :
Un forfait de 66 € auquel s'ajoute 0.25 € par kilomètre au-delà de 70 km.
Votre travail consiste à écrire un script qui permette de calculer automatiquement le coût C du contrat en fonction de la distance parcourue.

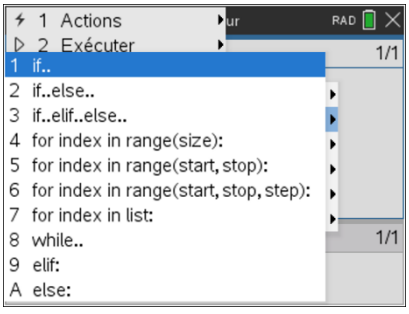
Langage naturel

X← *a*
Si (0<*X*) et (X<70)
    Alors *C* prend la valeur 66
    Sinon *C* prend la valeur 66 + 0.25*X*
Fin Si

Conseil à l'enseignant : Prévoir éventuellement le cas où l'utilisateur saisit un nombre *X* négatif.

Mise en œuvre :

- Démarrer l'application.
- Commencer un nouveau script Python et le nommer « TARIF ». Valider en appuyant sur la touche **enter**.
- Partager la fenêtre en deux parties (éditeur ; console) comme vu dans l'unité 1.
- Appuyer sur la touche menu puis choisir **4 Intégrés** et enfin **2 Contrôle**.
- Choisir le menu **3:if..elif..else…**



Unité 2 : Débuter la programmation en Python

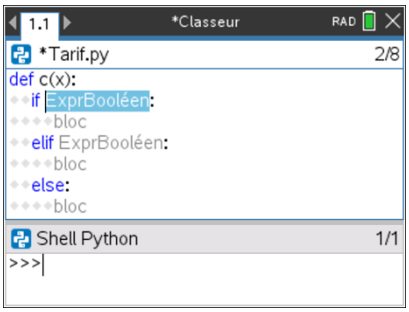
Compétence 1 : Instruction conditionnelle

Dans cette première leçon de l'unité 2, vous allez découvrir comment écrire et utiliser une instruction conditionnelle en Python.

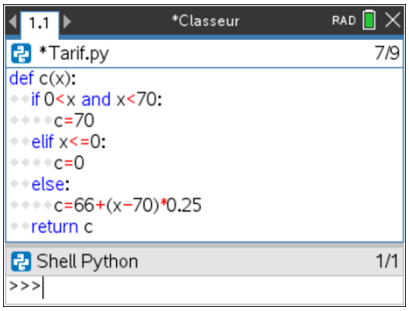
Objectifs :

- Écrire et utiliser une instruction conditionnelle.
- Réinvestir la notion de fonction en Python.

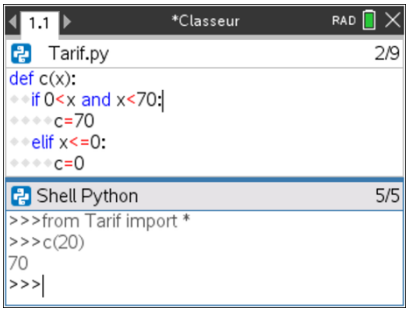
- Observer l'indentation du bloc conditionnel.
- Le compléter en utilisant l'algorithme proposé en langage naturel.
- Utiliser la touche [tab] afin d'accéder facilement à chaque bloc.



- Vous devriez obtenir le script ci-contre.
- L'instruction < et <= ainsi que and se trouve dans le menu **4 Intégrés** puis **3 Ops.** mais sont également accessibles en appuyant sur les touches [ctrl] **=**.



- Appuyer sur [ctrl] [B] afin d'enregistrer le script et de vérifier sa syntaxe.
- Exécuter le script [ctrl] [R] pour déterminer le coût d'un trajet avec une voiture louée à cette société.



Unité 2 : Débuter la programmation en Python

Compétence 1 : Instruction conditionnelle

Dans cette première leçon de l'unité 2, vous allez découvrir comment écrire et utiliser une instruction conditionnelle en Python.

Objectifs :

- Écrire et utiliser une instruction conditionnelle.
- Réinvestir la notion de fonction en Python.

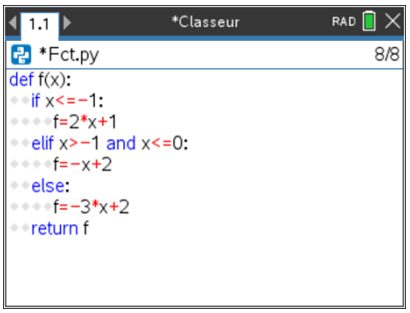
Conseil à l'enseignant : ctrl B sauvegarde le script, mais n'enregistre pas le classeur. Appuyer sur doc afin d'enregistrer le classeur dans un dossier.

Appliquons nos connaissances : Fonction par morceaux

On considère la fonction affine par morceaux f définie par :

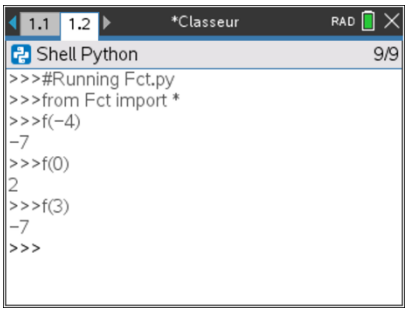
$$f(x) = \begin{cases} 2x + 1 \; si \; x \leq -1 \\ -x + 2 \; si \; x \in]-1\,;0] \\ -3x + 2 \; si \; x > 0 \end{cases}$$

Copier le script ci-contre et l'exécuter afin de compléter le tableau ci-dessous :



x	-4	-1.5	-0.5	-0.1	0.6	2.5	4.8	7.3
$f(x)$								

Exécuter le script.



Unité 2 : Débuter la programmation en Python

Compétence 2 : La boucle bornée For

Dans cette première leçon de l'unité 2, vous allez découvrir comment écrire et utiliser une instruction conditionnelle en Python.

Objectifs :

- Découvrir et mettre en oeuvre la boucle bornée FOR.
- Utiliser la boucle FOR dans des exemples simples.

Il est parfois utile dans un programme de répéter une ou plusieurs instructions un nombre défini de fois. Si le nombre de répétition du processus est connu à l'avance, on utilise une boucle bornée **For**.

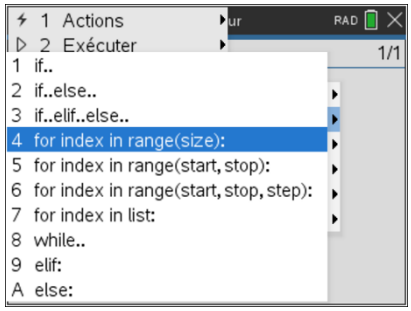
La syntaxe d'une boucle For est la suivante :

Langage naturel

Pour variable **allant de** minimum **à** maximum
    Instructions

Langage Python

for variable in range ( ) :
    Instructions



La fonction range( ) permet d'énumérer le nombre de passages dans la boucle bornée. Elle peut être appelée de plusieurs façons :

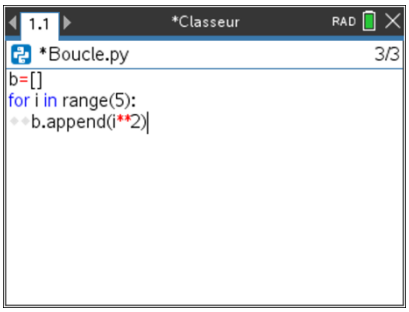
- for i in range(taille) : prend les valeurs entière de 0 à $taille - 1$, donc "$taille$" valeurs.
- for i in range(début, fin) : la variable i prend des valeurs entières de $début$ à $fin - 1$, où $début$ et fin sont ici des entiers.
- for i in range(début, fin, pas) : la variable i prend des valeurs entières de $début$ à $fin - 1$ par valeurs s'incrémentant de 1. $début$, sont ici des entiers.
- for i in liste : la variable i utilisera directement les valeurs de la liste de la première valeur jusqu'à la dernière.

Il n'existe pas d'instruction de fin de boucle. C'est l'indentation, c'est-à-dire le décalage vers la droite d'une ou plusieurs lignes, qui permet de marquer la fin de la boucle.

Mise en œuvre :

Vous allez créer un script permettant de bien comprendre ce qu'est une boucle, ainsi qu'un processus d'itération.

- Commencer un nouveau script et le nommer « BOUCLE »
- Initialiser une liste vide avec l'instruction b=[ ]. En langage Python, les éléments d'une listes sont placés entre [ ], séparés par des virgules.
- Appuyer sur menu et choisir dans le menu **4 Intégrés**, puis **2 Contrôle**, l'option **4 : for index in range(size) :**
- La méthode **.append( )** permet de compléter une liste. Ainsi, **b.append(i**2)** incrémente la liste **b** des carrés de *i*. A chaque valeur de *i*, la valeur de i^2 est placée en fin de liste.
- La variable i varie de 0 à 4 ce qui correspond bien à un éventail de **5** valeurs.



Unité 2 : Débuter la programmation en Python

Compétence 2 : La boucle bornée **For**

Dans cette première leçon de l'unité 2, vous allez découvrir comment écrire et utiliser une instruction conditionnelle en Python.

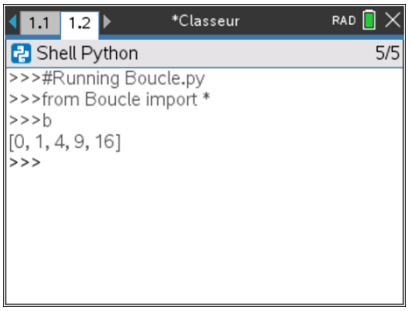
Objectifs :

- Découvrir et mettre en oeuvre la boucle bornée FOR.
- Utiliser la boucle FOR dans des exemples simples.

Conseil à l'enseignant : Attention, les compteurs de boucles sont toujours initialisés à 0.
La méthode **.append** concerne les listes. Pour l'atteindre, appuyer sur **4 Intégrés** puis choisir le menu **4 Listes** et enfin choisir **6 : .append( )**

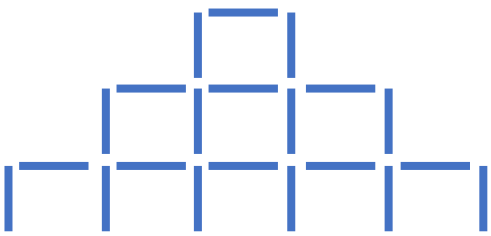
- Appuyer sur la touche ctrl B afin de sauvegarder le script et vérifier sa syntaxe.
- Appuyer sur la touche ctrl R pour exécuter le script.

Demander ensuite l'affichage des valeurs de **b** à partir de la console.



Conseil à l'enseignant : Dans une boucle ou une instruction comportant une indentation, toute écriture indentée d'une commande fait partie de la boucle. La fin de la boucle est marquée par la sortie de l'indentation.

Appliquons nos connaissances :
On réalise une construction à base de bâtons.
La première rangée notée « rangée 0 » est formée de 3 bâtons, la seconde de 7 bâtons et la troisième rangée de 11 bâtons.



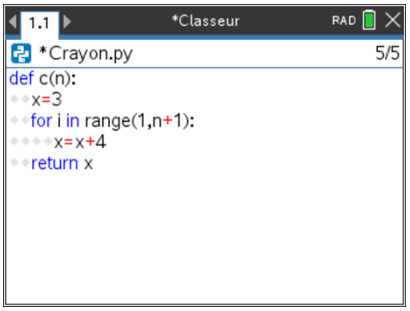
Rangée n°0

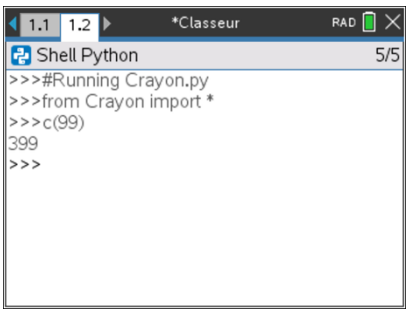
Rangée n°1

Rangée n°2

De combien de bâtons sera formée la rangée n°4 ?
Réaliser le script, puis l'exécuter

- Exécuter le programme pour déterminer combien la 100ème rangée comptera de bâtons.





Conseil à l'enseignant : Appuyer sur la touche **VAR** lors de l'exécution d'un script afin de rappeler les variables ou fonctions écrites dans celui-ci.

Unité 2 : Débuter la programmation en Python

Compétence 3 : La boucle non bornée **While**

Dans cette troisième leçon de l'unité 2, vous allez découvrir comment répéter un processus ou un ensemble d'instructions en utilisant une boucle non bornée **While**.

Objectifs :

- Découvrir et mettre en oeuvre la boucle non bornée **While**.
- Utiliser la boucle **While** dans des exemples simples.

Il est parfois utile dans un programme de répéter une ou plusieurs instructions un nombre indéfini de fois. Si le nombre de répétition du processus n'est pas connu à l'avance, on utilise une boucle non bornée **While**.
La boucle est alors parcourue autant de fois que nécessaire jusqu'à ce qu'une condition ne soit plus vérifiée. Tant que cette condition est vérifiée, la boucle continue.

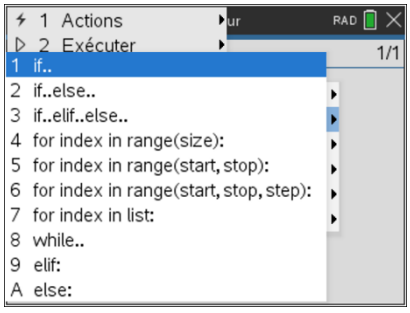
La syntaxe d'une boucle **While** est la suivante :

Langage naturel

Tant que condition **faire**
    Instructions

Langage Python

while condition :
    Instructions



Il n'existe pas d'instruction de fin de boucle. C'est l'indentation, c'est-à-dire le décalage vers la droite d'une ou plusieurs lignes, qui permet de marquer la fin de la boucle.

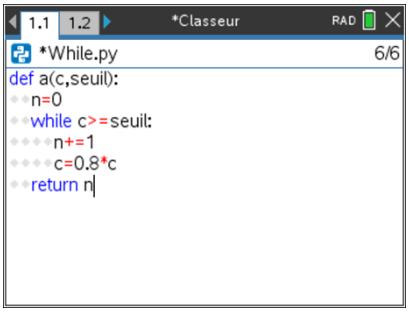
Mise en œuvre : Exemple n°1

Vous allez créer un script permettant de bien comprendre ce qu'est une boucle ainsi qu'un processus d'itération.
Mettre en œuvre l'algorithme ci-dessous :

L'algorithme a pour objet de déterminer le plus petit entier n tel que le terme général de la suite récurrente définie par $c_0 = 3{,}4$ et $c_{n+1} = 0{,}8\ c_n$ est strictement inférieur à 1.

$n \leftarrow 0$
$c \leftarrow 3.4$
Tant que $c \geq 1$
$n \leftarrow n+1$
$c \leftarrow 0.8c$
Fin Tant que

- Commencer un nouveau script et le nommer « WHILE ».
- L'instruction **While** est accessible en tapant sur menu puis **4 Intégrés** et enfin **2 Contrôle** puis en choisissant l'instruction **8 while condition :**
- Tant que la variable ***c*** sera strictement supérieure au seuil la variable *n* sera incrémentée de 1.



Unité 2 : Débuter la programmation en Python

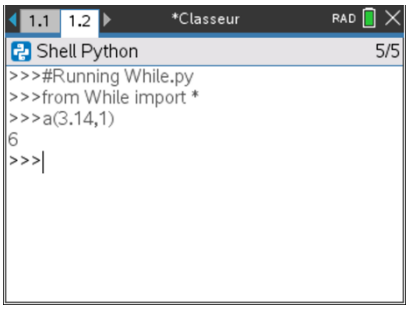
Compétence 3 : La boucle non bornée **While**

Dans cette troisième leçon de l'unité 2, vous allez découvrir comment répéter un processus ou un ensemble d'instructions en utilisant une boucle non bornée **While**.

Objectifs :

- Découvrir et mettre en oeuvre la boucle non bornée **While**.
- Utiliser la boucle **While** dans des exemples simples.

- Sauvegarder votre script ctrl B.
- Appuyer sur ctrl R pour exécuter le script et observer l'affichage.



Appliquons nos connaissances : Les rebonds du ballon.

Une balle est lâchée d'une hauteur de 1,20 m et rebondit sur le sol des $\frac{3}{5}$ de la hauteur du rebond précédent.
Vous devez élaborer un algorithme donnant le nombre de rebonds au bout duquel la hauteur atteinte par la balle est strictement inférieure à 1 cm.

- Définir une variable *H* dans laquelle on stockera la hauteur du rebond exprimée en cm et *R* une variable pour compter les rebonds.
- On devra répéter plusieurs fois l'instruction « *H* prend la valeur $\frac{3}{5} \times H$ » sans connaitre à l'avance le nombre de répétitions.
- On testera donc si $H \geq 1$ et le traitement dans la boucle sera réalisé tant que cette condition restera vérifiée.

Algorithme

H ← 90
R ←0
Tant que $H \geq 1$
| H← $\frac{3}{5} \times H$
| R← R+1
Fin Tant que

Unité 2 : Débuter la programmation en Python

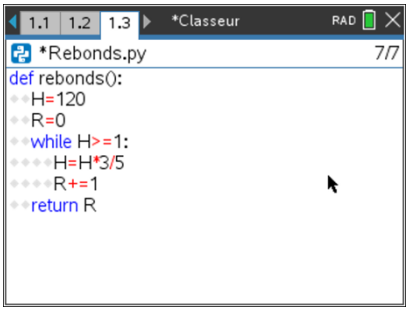
Compétence 3 : La boucle non bornée **While**

Dans cette troisième leçon de l'unité 2, vous allez découvrir comment répéter un processus ou un ensemble d'instructions en utilisant une boucle non bornée **While**.

Objectifs :

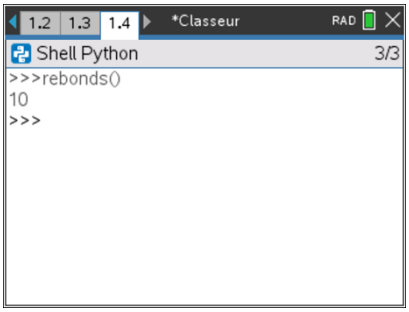
- Découvrir et mettre en oeuvre la boucle non bornée **While**.
- Utiliser la boucle **While** dans des exemples simples.

- Créer un nouveau script et le nommer Rebonds.
- On propose d'utiliser une fonction. Ainsi on veillera à respecter les indentations (une pour l'instruction **While** et une seconde pour l'instruction **return** permettant de renvoyer le contenu de la variable R.
- Appuyer sur [menu] et choisir le menu **3 : Modifier** puis **2 Désindentation(Maj+Tab)** afin de fermer la boucle **While**, tout en conservant l'exécution de la fonction.



- Sauvegarder votre script [ctrl] [B].
- Appuyer sur [ctrl] [R] pour exécuter le script.
- Entrer le nom de la fonction « rebonds() » et valider en appuyant sur la touche **enter**.
- Enrichir éventuellement le script à l'aide d'un message, par exemple :

return (« Nombre de rebonds = »,R)



Conseil à l'enseignant : Il peut s'avérer nécessaire de devoir conserver les valeurs, lorsque celles-ci correspondent à un calcul qui doit être par exemple réutilisé ou simplement conservé afin d'être comparé (suite numérique par exemple). Dans ce cas l'utilisation des listes est préconisée.

Un défi : calculer la distance totale parcourue par la balle jusqu'à son immobilisation. (on supposera que les rebonds de la balle sont rigoureusement situés sur la verticale)

Unité 2 : Débuter la programmation en Python

Application : Boucles et tests

Pour cette application de l'unité 2, on se propose de réinvestir les notions vues dans les leçons concernant les instructions conditionnelles ainsi que les boucles bornées et non bornées.

Objectifs :

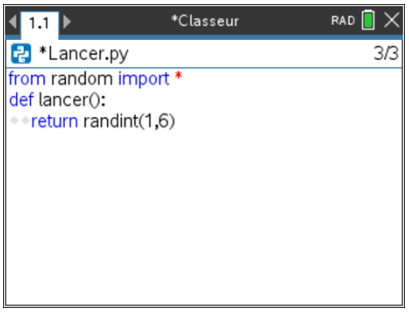
- Utiliser la boucle **While** et **For** pour mettre en oeuvre un algorithme relatif à un problème de probabilités ou de statistiques.

Dans cette application, vous allez écrire un script permettant de :

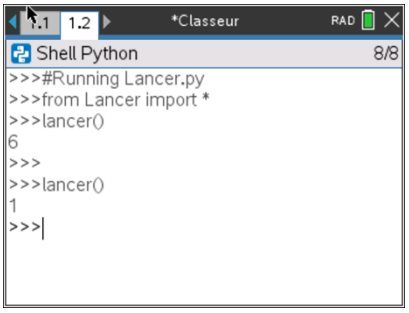
- Obtenir un nombre aléatoire en créant une fonction **lancer**.
- Utiliser cette fonction dans un autre script afin de déterminer le nombre de lancers nécessaires pour obtenir une somme de 12 lors du lancer de 2 dés à 6 faces parfaitement équilibrés.
- Lors du lancer d'un seul dé à 6 faces, obtenir le nombre de fois où chaque face apparait afin éventuellement de calculer une fréquence à comparer à la probabilité de sortie de chaque face.

Lancer un dé.

- Le travail sur les nombres aléatoires nécessite le chargement du module **random** avant la définition de la fonction.
- Créer un nouveau script et le nommer **Lancer**.
- Appuyer sur menu puis **6 Nombres aléatoires** et choisir le module **1 from random import***.
- Définir la fonction **lancer()** permettant d'obtenir un nombre aléatoire entier entre 1 et 6.



- Tester votre script après l'avoir vérifié.
- Utiliser les touches de direction vers le haut (▲) afin de relancer le script.



Unité 2 : Débuter la programmation en Python

Application : Boucles et tests

Pour cette application de l'unité 2, on se propose de réinvestir les notions vues dans les leçons concernant les instructions conditionnelles ainsi que les boucles bornées et non bornées.

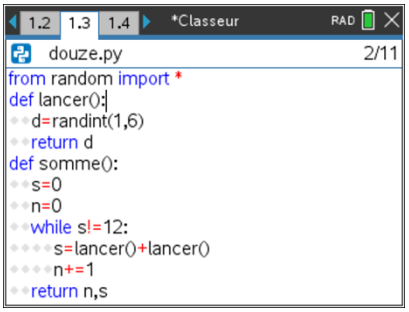
Objectifs :

- Utiliser la boucle **While** et **For** pour mettre en oeuvre un algorithme relatif à un problème de probabilités ou de statistiques.

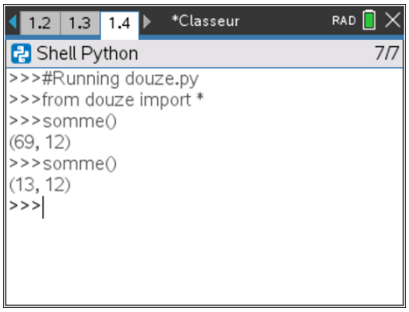
Nombre d'essais nécessaires.

On lance deux dés à 6 faces parfaitement équilibrés et on additionne les deux résultats obtenus. Vous allez écrire un autre script afin de modéliser les lancers de ces deux dés et rechercher le nombre *n* de lancers nécessaires afin d'obtenir la valeur 12.

De nombreuses solutions sont possibles, mais la première qui vient à l'esprit est de réutiliser la fonction précédente.



- La variable *s* donne la somme des lancers et *n* le nombre de lancers nécessaires avant d'atteindre 12.
- Toutes deux sont initialisées à 0.
- En langage Python, le symbole ≠ est représenté par **!=**. Ce symbole est obtenu en appuyant sur les touches menu puis **4 Intégrés** et enfin **3 Ops**.
- Compléter le script en veillant à respecter l'indentation puis l'exécuter.
- Le premier nombre donne le nombre de lancers nécessaires pour atteindre la somme 12 affichée par le second.



Pour cette application de l'unité 2, on se propose de réinvestir les notions vues dans les leçons concernant les instructions conditionnelles ainsi que les boucles bornées et non bornées.

Objectifs :

- Utiliser la boucle **While** et **For** pour mettre en oeuvre un algorithme relatif à un problème de probabilités ou de statistiques.

Échantillonnage et fréquence

Vous venez d'observer sur l'exemple précédent que le nombre d'essais nécessaires avant d'obtenir un 12 fluctue. On peut donc être conduit à calculer la fréquence d'échantillonnage, qui pour un grand nombre d'essais doit tendre vers la probabilité théorique.

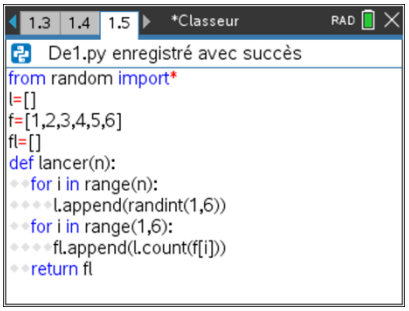
Vous allez dans ce dernier script :

Utiliser une boucle for pour répéter un seul lancer.

Calculer le nombre de fois où une face apparait.

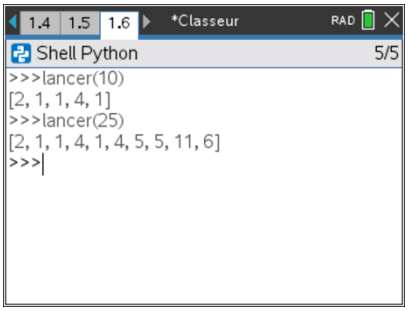
Utiliser une liste, ce qui a l'avantage de rendre le script plus court.

- Créer un nouveau script et le nommer De1.
- Le résultat d'un lancer est stocké dans la liste l précédemment initialisée vide par l'instruction **l=[ ]**.
- La liste **f** contient les numéros des faces.
- Ensuite un test est effectué sur la valeur de la variable (1 à 6) et chaque fois qu'une condition est vérifiée, une nouvelle liste **fl** est créée dans laquelle est stocké, pour chaque numéro de face variant de 1 à 6, le nombre d'occurrence observées.



- Modifier éventuellement le script afin de calculer la fréquence d'apparition de chaque face.

- Appuyer sur la touche ctrl B pour enregistrer le script et vérifier sa syntaxe.
- Appuyer sur la touche ctrl R afin d'exécuter le script. Observer les effectifs de sortie de chaque face, pour 100 lancers : les fréquences sont simples à calculer.



Unité 3 : Débuter la programmation en Python

Compétence 1 : Fonctions et boucles

Dans cette première leçon de l'unité 3, vous allez mettre en oeuvre vos connaissances en algorithmique et en langage Python afin de :

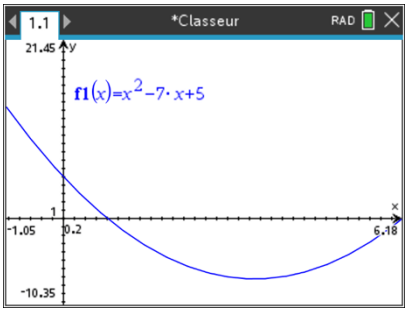
- Rechercher les solutions d'une équation $f(x)=0$.
- Résoudre un problème d'optimisation.

Objectifs :

- Utiliser une fonction en langage Python.
- Mettre en oeuvre la boucle bornée **While**.

Principe de la dichotomie

On considère la fonction *f* définie sur l'intervalle $[-2,3]$ par $f(x) = x^2 - 7x + 5$.
On utilise la calculatrice afin de tracer la courbe C_f représentant les variations de la fonction f.
Vous allez résoudre l'équation $f(x) = 0$ en écrivant un script Python correspondant à un algorithme connu et appelé « algorithme de dichotomie ».



Pour comprendre ce qu'est la dichotomie, on propose une petite expérience : « *chercher un mot dans un gros dictionnaire papier de 1024 pages* »

- Vous l'ouvrez au milieu : le mot ne s'y trouve pas, mais il est avant (il est donc dans les 512 premières pages).
- Vous ouvrez la moitié de la 1ère moitié : le mot ne s'y trouve pas, mais il est après (il est donc entre les pages 257 et 512).
- Vous ouvrez la moitié de la 2ème moitié… etc.

A chaque fois que vous progressez, le nombre de pages qui reste à examiner est divisé par 2.

Ainsi, dans un dictionnaire de 1024 pages, vous êtes certain de trouver votre page en 10 recherches seulement, puisque $1024/(2^{10})=1$

Unité 3 : Débuter la programmation en Python

Compétence 1 : Fonctions et boucles

Dans cette première leçon de l'unité 3, vous allez mettre en oeuvre vos connaissances en algorithmique et en langage Python afin de :

- Rechercher les solutions d'une équation $f(x)=0$.
- Résoudre un problème d'optimisation.

Objectifs :

- Utiliser une fonction en langage Python.
- Mettre en oeuvre la boucle bornée **While**.

Algorithme :

Tant que $b-a > prec$ faire :
  $m \leftarrow \frac{a+b}{2}$
  Si $f(m)$ et $f(a)$ sont de signes, opposés
    $b \leftarrow m$
  sinon
    $a \leftarrow m$
Fin Tant que

Commentaires :

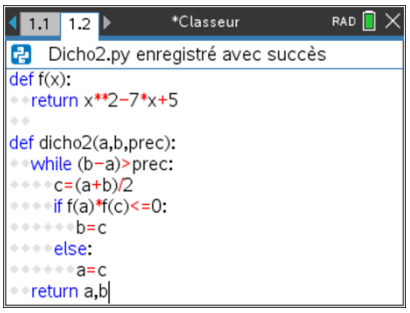
[a,b] : Bornes de l'intervalle d'étude
f : Fonction étudiée, continue sur $[a\ ;\ b]$.
On se place au milieu de l'intervalle $[a\ ;\ b]$.
Si $f(m)$ *et* $f(a)$ *sont de même signe, alors la solution de l'équation* $f(x) = 0$ *est située dans l'intervalle* $[m\ ;\ b]$.
On se place donc sur $[m\ ;\ b]$
sinon.
on se place sur $[a\ ;m]$.

Mise en œuvre de l'algorithme :

- Encadrer entre deux entiers la valeur x_0 solution de l'équation $f(x) = 0$ avec une précision donnée « *prec* ».
- Vous remarquerez que $f(0) \times f(1) < 0$
- Calculer $f(0) \times f\left(\frac{1}{2}\right)$ et $f\left(\frac{1}{2}\right) \times f(1)$
- En déduire si x_0 appartient à l'intervalle $\left[0\ ;\frac{1}{2}\right]$ ou $\left[\frac{1}{2}\ ;1\right]$

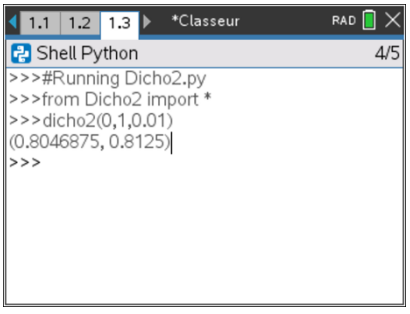
Créer un nouveau script et le nommer DICHO

- Entrer les différentes instructions, celles-ci se trouvent pour leur ensemble sous l'onglet (menu puis **4 Intégrés**).
- Les tests peuvent être obtenus directement par l'appui sur les touches menu puis **4 Intégrés** et enfin **3 Ops**.
- $[a, b]$ représente l'intervalle d'étude et n le nombre d'étapes.



Exécuter le script :

A partir de la représentation graphique de la fonction, tester le script.



Unité 3 : Débuter la programmation en Python

Compétence 1 : Fonctions et boucles

Dans cette première leçon de l'unité 3, vous allez mettre en oeuvre vos connaissances en algorithmique et en langage Python afin de :

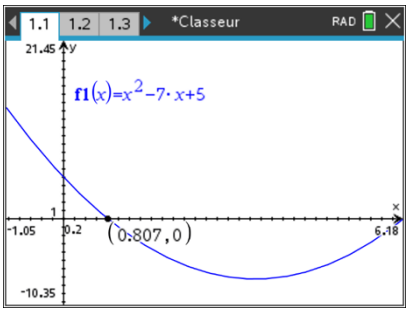
- Rechercher les solutions d'une équation $f(x)=0$.
- Résoudre un problème d'optimisation.

Objectifs :

- Utiliser une fonction en langage Python.
- Mettre en oeuvre la boucle bornée **While**.

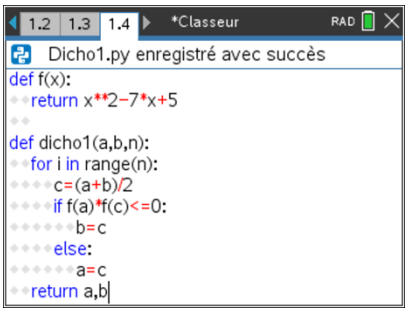
Affiner votre recherche afin de trouver une solution proche de celle qui est proposée par la calculatrice.

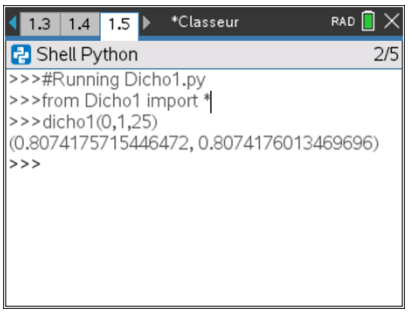
La valeur exacte de x_0 dans l'intervalle [0 ; 1] étant donnée par : $x_0 = \frac{7-\sqrt{29}}{2}$



Prolongements possibles :

a) Au lieu de donner la précision, on peut travailler avec le nombre d'étapes.





La suite du script reste identique

b) Envisager une programmation récursive

Remarque : Pour plus d'information sur la récursivité voir Compétence 3 de l'unité 3

```
def dicho(a,b,prec) :
    if (b-a)<=prec :
        return a,b
    else :
        c=(a+b)/2
        if f(a)*f(b)<=0 :
            return dicho(a,c,prec)
        else :
            return dicho(c,b,prec)
```

Unité 3 : Débuter la programmation en Python — Compétence 2 : La boucle bornée For

Dans cette seconde leçon de l'unité 3, vous allez découvrir comment répéter un processus ou un ensemble d'instructions en utilisant une boucle bornée **FOR**.

Objectifs :

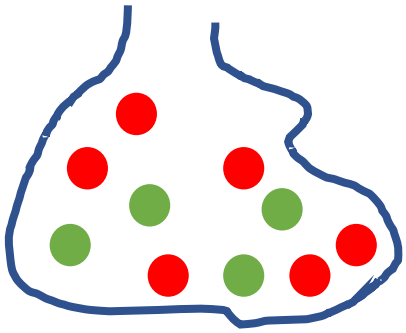
- Appliquer une fonction.
- Découvrir et mettre en oeuvre la boucle bornée **FOR**.
- Utiliser la boucle **FOR** dans des exemples simples.

Constituer un échantillon :

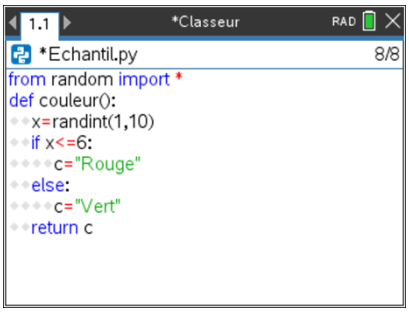
Un sac opaque contient six jetons rouges et quatre jetons verts. On tire au hasard un jeton du sac, on note sa couleur puis on le replace dans le sac.

Programmer une fonction **couleur()** simulant cette expérience aléatoire de variable X.

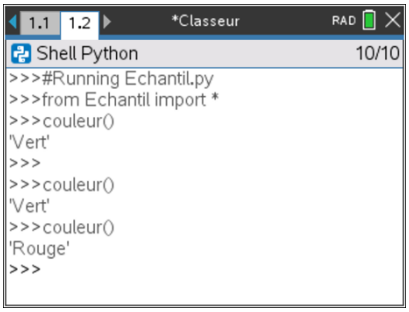
- Quelles sont les « valeurs » possibles prises par la variable X ?
- On souhaite écrire un script qui permette de distinguer les boules rouges des vertes à l'aide d'un test.



- Commencer un nouveau script et le nommer « **ECHANTIL** ».
- Comme vous travaillez sur des nombres aléatoires, le chargement de la bibliothèque « **random** » est nécessaire. Pour cela, appuyer sur menu puis **6 Nombres aléatoires**.
- Entrer le script ci-contre dans l'éditeur en veillant à respecter l'indentation.



- Afficher le résultat de la fonction dans la console.
- Exécuter le script ctrl R plusieurs fois en appelant la fonction **couleur()**



Conseil à l'enseignant : ce script peut être modifié pour être appliqué sur un nombre quelconque de jetons. Dans ce cas-là, on pourra définir **couleur(n,a)** où **n** est le nombre de jetons et **a** le nombre de jetons rouges.

Unité 3 : Débuter la programmation en Python — Compétence 2 : La boucle bornée **For**

Dans cette seconde leçon de l'unité 3, vous allez découvrir comment répéter un processus ou un ensemble d'instructions en utilisant une boucle bornée **FOR**.

Objectifs :

- Appliquer une fonction.
- Découvrir et mettre en oeuvre la boucle bornée **FOR**.
- Utiliser la boucle **FOR** dans des exemples simples.

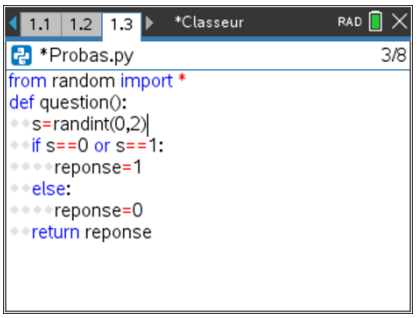
Appliquons nos connaissances : Échantillonnage et prise de décision.
On a réalisé un sondage à la sortie du nouveau spectacle proposé par un artiste. Ce sondage réalisé dans une grande ville montre que les deux tiers des personnes ayant vu le spectacle l'ont aimé. L'agent de l'artiste pense que toute la population française est dans le même état d'esprit. Il commande une enquête auprès d'un institut de sondage pour le vérifier.

Étude de la population :
Pour les besoins de l'enquête statistique, l'institut de sondage doit créer une fonction qui simule la réponse à la situation.
Votre travail consiste à créer cette fonction en respectant le cahier des charges suivant :

- Le spectacle a été apprécié, avec une probabilité $p = \frac{2}{3}$.
- Le spectacle n'a pas été apprécié, avec une probabilité $p = \frac{1}{3}$.

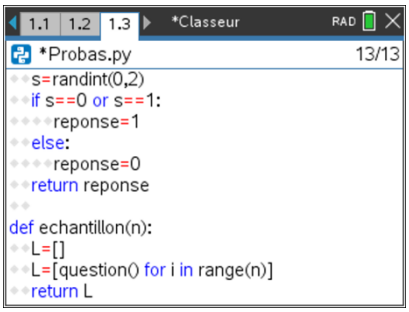
1. Commencer un nouveau script et le nommer **PROBAS**
2. Écrire cette fonction dans l'éditeur et la tester plusieurs fois en appuyant sur [ctrl] [R] puis en tapant dans la console le nom de la fonction sans arguments : **question( )**.

Astuce : on peut utiliser la touche [var] puis flèche du haut pour répéter l'exécution, (c'est bien plus rapide que de taper le nom de la fonction…).



Simulation d'un échantillon de taille **n** :
L'institut de sondage souhaite simuler des échantillons de taille variable.
Vous devez donc créer dans le script courant une fonction **échantillon(n)** permettant de poser la question à un échantillon de taille **n**.

- Pour cela, créer une liste vide **L**.
- Remplir cette liste en utilisant la fonction **question( )** et en utilisant une boucle **FOR**.



Conseil à l'enseignant : Le langage Python permet d'utiliser une fonction pour remplir une liste incrémentée par une boucle **FOR**, comme arguments de la liste.
La réalisation de cet exemple est possible sans l'utilisation de listes. Les scripts sont alors à modifier légèrement en remplaçant les instructions relatives aux listes par des boucles bornées incrémentant une variable.

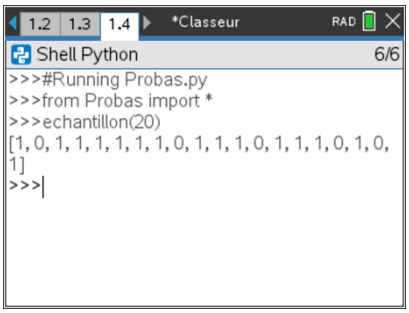
Unité 3 : Débuter la programmation en Python — Compétence 2 : La boucle bornée **For**

Dans cette seconde leçon de l'unité 3, vous allez découvrir comment répéter un processus ou un ensemble d'instructions en utilisant une boucle bornée **FOR**.

Objectifs :

- Appliquer une fonction.
- Découvrir et mettre en oeuvre la boucle bornée **FOR**.
- Utiliser la boucle **FOR** dans des exemples simples.

- Tester la fonction **echantillon(n)** pour un échantillon de taille 20, en exécutant le script et en tapant dans la console « echantillon(20) ».



Validation de l'échantillon :

L'institut de sondage souhaite déterminer le pourcentage d'échantillons dont la fréquence de personnes qui ont apprécié le spectacle appartient à l'intervalle de fluctuation à 95% de $p = \frac{2}{3}$

A la suite du script précédent, il vous est demandé de créer deux fonctions :

a) **freq_echantillon(n)** qui calcule automatiquement la fréquence des réponses 1 observée dans l'échantillon **L** de taille **n**.

b) **interv_fluctu(te, ne)** pour déterminer si la fréquence d'un échantillon de taille **n** est compris dans l'intervalle de fluctuation à 95%.

Algorithme

Fonction interv_fluctu(taille_ech , nbre_ech)
  Nf ← 0
  Pour *i* allant de 1 à nbre ech faire
    f← freq_echantillon(taille ech)
    Si *f* appartient à $\left[p - \frac{1}{\sqrt{n}} ; p + \frac{1}{\sqrt{n}}\right]$ alors
      $Nf \leftarrow Nf + 1$
    Fin si
  Fin Pour

Unité 3 : Débuter la programmation en Python

Compétence 2 : La boucle bornée For

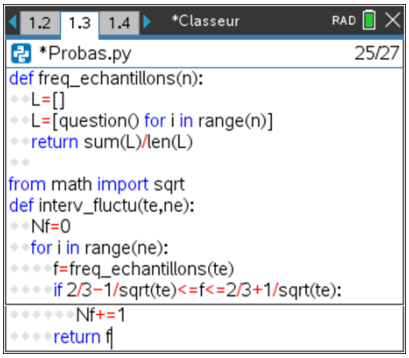
Dans cette seconde leçon de l'unité 3, vous allez découvrir comment répéter un processus ou un ensemble d'instructions en utilisant une boucle bornée **FOR**.

Objectifs :

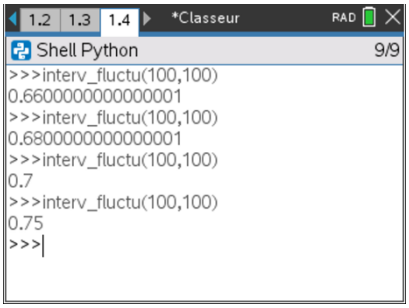
- Appliquer une fonction.
- Découvrir et mettre en oeuvre la boucle bornée **FOR**.
- Utiliser la boucle **FOR** dans des exemples simples.

Conseil à l'enseignant : On rappelle que pour un caractère dont la proportion dans une population donnée est *p*.
Pour $n \geq 25$ et $0.2 \leq p \leq 0.8$, la fréquence du caractère dans les échantillons de taille n appartient à l'intervalle $\left[p - \frac{1}{\sqrt{n}} ; p + \frac{1}{\sqrt{n}}\right]$ dans 95% des cas.
Cet intervalle est appelé intervalle de fluctuation à 95%.

A partir de la fonction *interv_fluctu* écrit en langage naturel, vérifier que vous obtenez la fonction interv_fluctu en Python.



- Appuyer sur [var] afin de rappeler la fonction **interv_fluctu**.
- Tester le script plusieurs fois pour un échantillon de taille 100.
- En déduire pour un exemple, l'intervalle de fluctuation à 95%.
- Cette étude statistique remet-elle en question l'affirmation de l'agent de l'artiste ?



Unité 3 : Débuter la programmation en Python

Compétence 3 : Programmation et récursivité

Dans cette troisième leçon de l'unité 3, vous allez utiliser les fonctions afin de réaliser une programmation récursive.

Objectifs :

- Appliquer une fonction.
- Découvrir et mettre en oeuvre la programmation récursive.

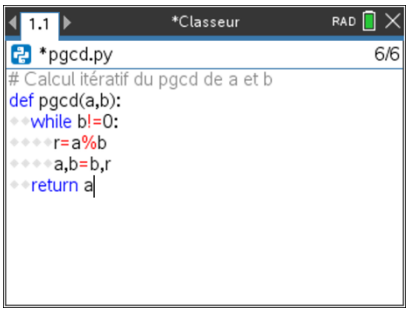
Calcul de PGCD (Méthode itérative)
Pour calculer le « plus grand commun diviseur de deux nombres entiers positifs » (PGCD), on peut utiliser l'algorithme d'Euclide.
Remarque : on suppose que $a > b$.

On procède de la manière suivante :

- On effectue la division euclidienne de *a* par *b*. On note *r* le reste (on n'utilise pas le quotient).
- On remplace ensuite *a* par *b* et *b* par *r*.
- **Tant que** le reste est différent de 0, on **réitère** le procédé.

Après un certain nombre d'itérations, on obtient un reste égal à 0.

Le PGCD de ***a*** et de ***b*** est alors le reste précédent (c'est à dire **le dernier reste non nul**).



- Créer un nouveau script et le nommer **pgcd**.
- Créer une fonction **pgcd(a,b)** en tapant menu **4 Intégrés** puis **1 Functions**.
- Le symbole ≠ s'écrit en langage Python à l'aide de ! = et se trouve dans le menu **4 Intégrés** puis **3 Ops**. Ou bien par la séquence ctrl =.
- Noter l'affectation **a,b=b,r** dans le script qui permet de gagner une ligne de code, mais qui correspond à la réalisation des affectations a = b et b = r.

- Exécuter le script pour différents couples de nombres entiers positifs.

Unité 3 : Débuter la programmation en Python | **Compétence 3 : Programmation et récursivité**

Dans cette troisième leçon de l'unité 3, vous allez utiliser les fonctions afin de réaliser une programmation récursive.

Objectifs :

- Appliquer une fonction.
- Découvrir et mettre en oeuvre la programmation récursive.

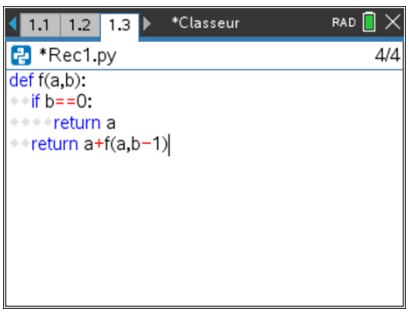
Un pas plus loin .

3 : Programmation récursive :
Un algorithme est dit récursif si, à un moment, il s'appelle lui-même.

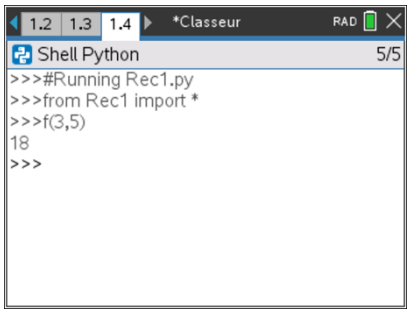
La récursivité peut posséder de nombreux avantages dans un algorithme. Premièrement, elle permet de résoudre des problèmes, d'habitude insolubles avec l'utilisation de simples boucles **Pour** ou **Tant que**. Elle peut aussi rendre un algorithme plus lisible et plus court, mais surtout, elle permet, dans certains cas, un gain colossal de temps comme c'est le cas avec les algorithmes de tri.

Un premier exemple de récursivité :

- Créer un nouveau script et le nommer **REC1**.
- Entrer le script ci-contre.
- Quel est le premier cas appliqué à cette fonction récursive ? (On rappelle que a et b sont des entiers positfs avec : $a > b$
- Qu'est ce qui garantit dans cette fonction récursive, que le script finira par s'arrêter ?
- Écrire le processus complet.



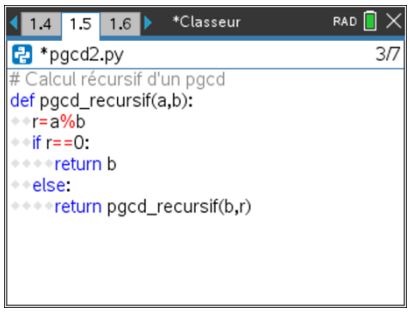
- Exécuter le script dans une console ctrl R.
- Que retourne *f(a,b)* *a* et *b* étant des entiers naturels non nuls ?



3.1 : Un calcul de pgcd récursif :

Le calcul du PGCD de deux entiers positifs *a* et *b* utilise toujours l'algorithme d'Euclide. Soit *r* le reste de la division euclidienne de *a* par *b* :
*a = b*q + r , r < b.*

Tout diviseur commun de *a* et *b* divise aussi *r = a – b*q*
et réciproquement tout diviseur commun de *b* et *r* divise aussi *a = b*q + r*.
Donc le calcul du PGCD de *a* et *b* se ramène à celui du PGCD de *b* et *r* ; et on peut recommencer sans craindre une boucle sans fin, car les restes successifs, entiers positifs forment une suite strictement décroissante. Le dernier reste non nul obtenu est le PGCD cherché.



Unité 3 : Débuter la programmation en Python

Compétence 3 : Programmation et récursivité

Dans cette troisième leçon de l'unité 3, vous allez utiliser les fonctions afin de réaliser une programmation récursive.

Objectifs :

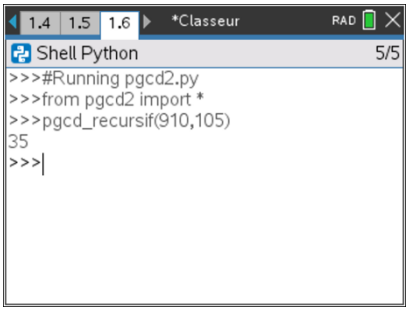
- Appliquer une fonction.
- Découvrir et mettre en oeuvre la programmation récursive.

Par exemple si *a*=96 et *b*=81, les calculs sont les suivants :

et le PGCD est 3.

a		*b*		*r*
96	= 1 *	81	+	15
81	= 5 *	15	+	6
15	= 2 *	6	+	3
6	= 2 *	3	+	0

- Exécuter le script en appuyant sur la touche ctrl R et le tester pour quelques valeurs.



Conseil à l'enseignant : Pour éviter les cercles vicieux, une fonction récursive doit toujours comporter un cas particulier où le résultat est calculé directement, c'est à dire sans appel récursif ; il faut aussi s'assurer que ce cas particulier finira toujours par se présenter.

Unité 3 : Débuter la programmation en Python

Application : Tests, boucles

Dans cette application de l'unité 3, vous allez utiliser les notions acquises dans les leçons précédentes afin de programmer des algorithmes vous permettant d'affiner vos connaissances des nombres et en particulier des nombres premiers.

Objectifs :

- Mettre en oeuvre les boucles et tests pour la programmation complète d'un algorithme en Python.

Un nombre entier naturel est dit premier s'il possède exactement deux diviseurs : 1 et lui-même.
Par exemple :

- 1 n'est pas premier (il ne possède qu'un seul diviseur : 1).
- 7 est un nombre premier (ses diviseurs sont 1 et 7).
- 8 n'est pas premier (il possède quatre diviseurs : 1, 2, 4 et 8).

La liste des nombres premiers sont : 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, …etc.
Il en existe une infinité.
On se propose de déterminer le 2020ème nombre premier.

On considère l'algorithme ci-contre où n est un entier naturel.

- Afin de comprendre l'algorithme, à quelle condition sur les nombres 2, 3, *n-1*, un entier $n \geq 2$ est-il premier ?
- Réaliser l'écriture de la fonction « **ep(*n*)** » qui à tout entier naturel ***n*** devra renvoyer 1 si ***n*** est premier et 0 sinon.

```
Si n ≤ 1 alors
    Retourner 0
Fin si
Pour k de 2 à n-1 Faire
    Si n% k = 0 Alors
        Retourner 0
    Fin si
Fin pour
Retourner 1
```

La partie principale du programme est donnée par l'algorithme ci-contre.
Votre travail consiste à implémenter cet algorithme en langage Python afin de répondre au problème posé.

```
N← 2
no← 1
Tant que no < 2020 Faire
    N← N + 1
    No← no + ep(n)
Fin Tant que
Afficher « Le 2020e nombre premier est », N
```

Unité 3 : Débuter la programmation en Python

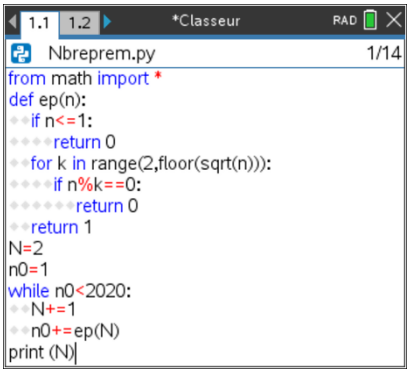
Application : Tests, boucles

Dans cette application de l'unité 3, vous allez utiliser les notions acquises dans les leçons précédentes afin de programmer des algorithmes vous permettant d'affiner vos connaissances des nombres et en particulier des nombres premiers.

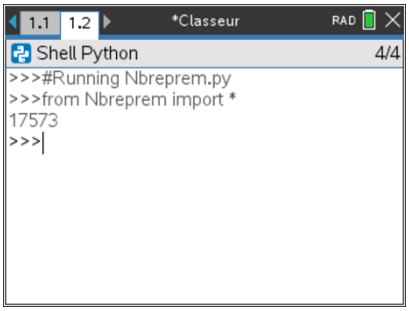
Objectifs :

- Mettre en oeuvre les boucles et tests pour la programmation complète d'un algorithme en Python.

- Commencer un nouveau script et le nommer **NBREPREM**.
- Entrer les différentes instructions en veillant à respecter l'indentation.



- Exécuter le script.
- Vérifier que le 2020e nombre premier est bien **17573**.



Conseil à l'enseignant : en changeant le test de primalité utilisé et en ne testant que sur 2 à la partie entière de : racine (n) +1 (ce qui assure de bien tester tous les diviseurs potentiels), on diminue considérablement le temps d'attente qui passe à quelques secondes.

Unité 4 : Utiliser la librairie ti_plotlib

Compétence 1 : Paramétrer une représentation

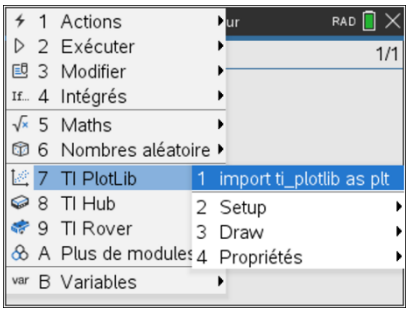
Dans cette première leçon de l'unité 4, vous allez découvrir comment écrire et utiliser une instruction permettant de faire des représentations graphiques en Python. Vous apprendrez également à représenter un graphique et paramétrer l'affichage.

Objectifs :

- Découvrir le module **ti_plotlib**.
- Représenter un point, un segment.
- Paramétrer une représentation graphique.

1 : La librairie ou module ti_plotlib

Pour effectuer une représentation graphique lors de l'exécution d'un script, celui-ci doit être en mesure de comprendre les instructions graphiques. Il est donc nécessaire « d'embarquer » les fonctions graphiques au sein d'une bibliothèque **TI PlotLib**.

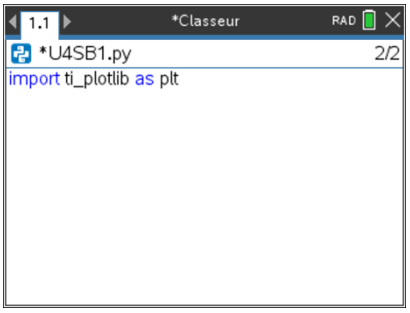


Commencer un nouveau script en le nommant U4SB1 et inclure dans celui-ci le module **TI PlotLib** (touches : menu Choisir le menu 7 **: TI PlotLib…** puis le menu **1 : import ti_plotlib as plt**.

Pour cette première partie, vous allez écrire un script permettant d'afficher un point dont les coordonnées sont connues. Ensuite, vous modifierez votre script afin de localiser votre point dans un repère et modifierez sa couleur.

Pour terminer cette première leçon, vous afficherez le nom de chaque axe et donnerez un titre à votre graphique.

Commencer un nouveau script en le nommant U4SB1 et inclure dans celui-ci le module ti-plotlib (touches : menu Choisir le menu 7 **: TI PlotLib…** puis le menu **1 : import ti_plotlib as plt**.

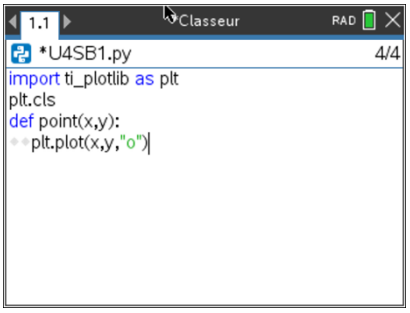


Pour cette première partie, vous allez écrire un script permettant d'afficher un point dont les coordonnées sont connues. Ensuite, vous modifierez votre script afin de localiser votre point dans un repère et modifierez sa couleur.

Pour terminer cette première leçon, vous demanderez l'affichage du nom de chaque axe et donnerez un titre à votre graphique.

Définir une fonction ayant pour argument les coordonnées d'un point, puis demander l'affichage de ce point.

- Dans un premier temps, nettoyez votre écran en utilisant l'instruction **plt.cls()** que vous trouverez dans le module **TI PlotLib** au menu **Configurer** (**2 : cls()**).
- Pour dessiner le point, choisir l'instruction **6 : Tracé un point**, située dans le menu **Dessin** du **module TI PlotLib**.
- Choisir la marque désirée.



Unité 4 : Utiliser la librairie ti_plotlib

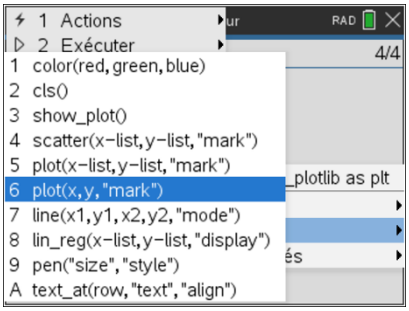
Compétence 1 : Paramétrer une représentation

Dans cette première leçon de l'unité 4, vous allez découvrir comment écrire et utiliser une instruction permettant de faire des représentations graphiques en Python. Vous apprendrez également à représenter un graphique et paramétrer l'affichage.

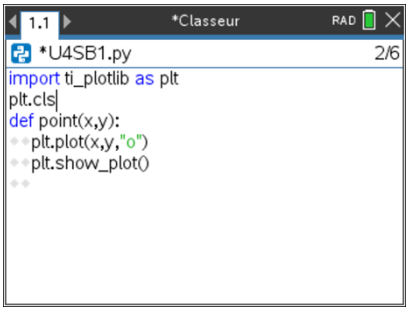
Objectifs :

- Découvrir le module **ti_plotlib**.
- Représenter un point, un segment.
- Paramétrer une représentation graphique.

Conseil à l'enseignant : La représentation d'un point sous forme de pixel est à choisir dans le cas où un grand nombre de points est à représenter.



Terminer le script en demandant l'affichage de la représentation par le choix de l'instruction **show_plot() (instruction n°3 du module Ti PlotLib)**.



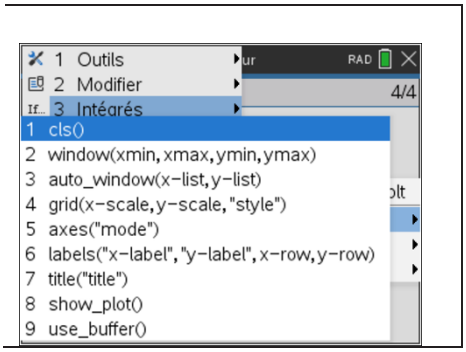
- Demander l'exécution du script ctrl R, puis appuyer sur la touche var afin de rappeler dans la console la fonction **point( )**.
- Donner les coordonnées d'un point et observer votre écran.
- Appuyer sur la touche del pour sortir de l'écran graphique, puis sur var afin de retrouver la liste des variables du script.
- Modifier les coordonnées de votre point (par exemple **point(10,10)**) et constater que celui-ci n'est plus visible à l'écran.

Conseil à l'enseignant : Lors de l'écriture d'un script utilisant les fonctions graphiques, il sera nécessaire de préciser les paramètres de la fenêtre graphique et éventuellement d'afficher un repère, grille, nom des axes… etc.

2 : Affiner votre représentation

Inclure l'instruction **plt.cls( )** sous la définition de la fonction, afin d'éviter d'avoir d'autres informations superposées à votre représentation graphique.

A partir des différentes options du menu **Setup** du module **TI PlotLib**, rajouter dans votre script les instructions permettant :



Unité 4 : Utiliser la librairie ti_plotlib

Compétence 1 : Paramétrer une représentation

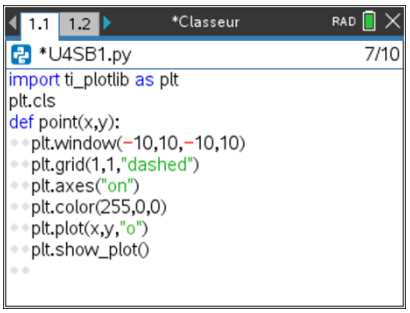
Dans cette première leçon de l'unité 4, vous allez découvrir comment écrire et utiliser une instruction permettant de faire des représentations graphiques en Python. Vous apprendrez également à représenter un graphique et paramétrer l'affichage.

Objectifs :

- Découvrir le module **ti_plotlib**.
- Représenter un point, un segment.
- Paramétrer une représentation graphique.

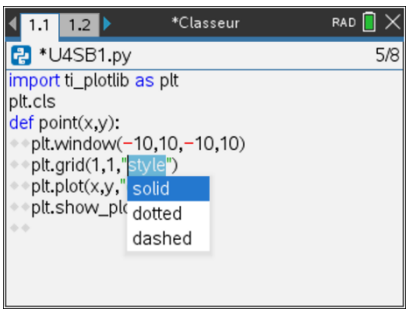
Conseil à l'enseignant : Pour couper, copier ou coller une ligne, utiliser les Outils (ctrl C ; ctrl V) à partir de l'éditeur de script

- De définir une fenêtre graphique telle que : X_{min} = -10 ; X_{max} = 10 ; Y_{min} = -10 et Y_{max} = 10 (**instruction 2 : window**).
- Afficher une grille (instruction **4 : grid**) ; le type de grille est laissé à votre choix.
- Afficher les axes (instruction **5 : axes**).
- Modifier la couleur du point (Menu **Draw,** puis **1 : color(red, green, blue)**).

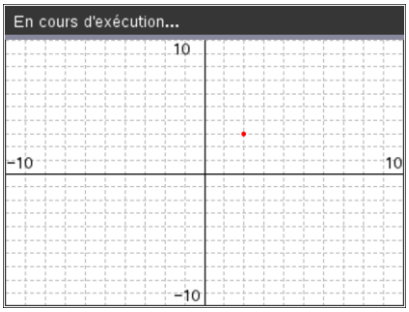


Conseil à l'enseignant : La couleur d'un point ou d'un tracé est à préciser en code **r, v, b** (rouge, vert, bleu), chaque paramètre pouvant prendre une valeur entière dans l'intervalle [0 ; 255]. Les couleurs sont codées sur 8 bits, soit 2^8 = 256 possibilités, en comptant le 0 qui correspond à une absence de la composante **r** ou **v** ou **b.**
Il est également possible de tracer la grille en couleur en complétant l'instruction **grid(xscl, yscl, « style », (r,v,b)).**

Utiliser la touche tab afin de compléter aisément les différents blocs. Une aide contextuelle est proposée afin de choisir comme ci-dessous, un style de représentation pour un point, une grille…etc.



Exécuter votre script et observer les changements. Vous devriez obtenir un écran identique à celui ci-contre.



Conseil à l'enseignant : Par ailleurs. Lors de l'exécution d'un script, la console est réinitialisée. L'historique est accessible en utilisant les touches de direction de la calculatrice. Mais attention, cet historique est perdu lors d'une nouvelle réinitialisation.

Modifier à nouveau le script afin de donner un nom à vos axes. Par exemple (« abscisse » et « ordonnée »).

Unité 4 : Utiliser la librairie ti_plotlib

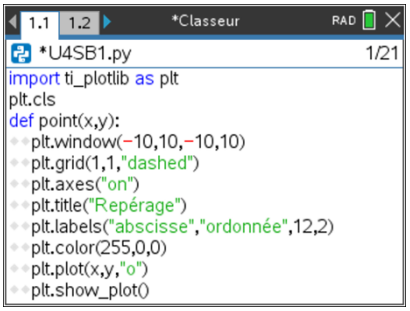
Compétence 1 : Paramétrer une représentation

Dans cette première leçon de l'unité 4, vous allez découvrir comment écrire et utiliser une instruction permettant de faire des représentations graphiques en Python. Vous apprendrez également à représenter un graphique et paramétrer l'affichage.

Objectifs :

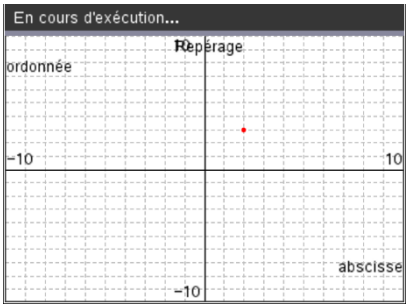
- Découvrir le module **ti_plotlib**.
- Représenter un point, un segment.
- Paramétrer une représentation graphique.

Pour cela, inclure dans votre script (peu importe l'emplacement) une ligne **plt.labels**. Celle-ci se trouve à l'emplacement **7 : labels()** du menu **Setup** dans le module **TI PlotLib**.



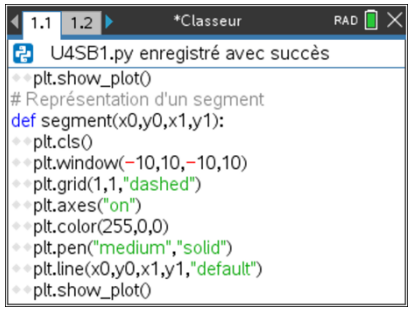
Conseil à l'enseignant : l'instruction **labels(« x-étiq », « y-étiq »,x ,y)** nommera les axes en plaçant les étiquettes aux lignes et colonnes x et y par défaut, celles-ci sont placées en ligne 12 pour x et 2 pour y, respectivement justifiées à gauche et à droite.

Si vous le souhaitez, vous pouvez également ajouter un titre à l'aide de l'instruction **plt.title(« Repérage »)**.



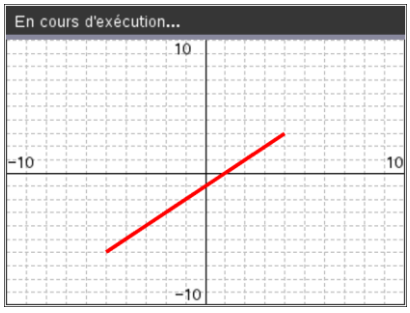
Rappel : Les caractères accentués sont obtenus à partir de l'éditeur de script en appuyant sur la touche ctrl, puis en utilisant les touches de direction afin de choisir dans la palette le caractère souhaité.

Pour aller plus loin : Compléter votre script en écrivant une fonction vous permettant de représenter graphiquement un segment. Vous trouverez ci-contre les instructions essentielles. Vous pouvez bien entendu si vous le souhaitez modifier le script afin d'afficher les axes, une grille ...etc.
L'option **plt.pen** accessible via le menu Dessin (**9 : pen(« size », « style »)**), permet de paramétrer la largeur du trait.



Conseil à l'enseignant : A partir de l'éditeur de script, vous pouvez écrire un commentaire. Celui-ci est écrit en gris et précédé d'un **#**, signifiant que cette instruction ne sera pas exécutée.
Le caractère # peut également être atteint en utilisant la touche ?!.

Exécuter votre script.



Unité 4 : Utiliser la librairie ti_plotlib — Compétence 2 : Représenter graphiquement une fonction

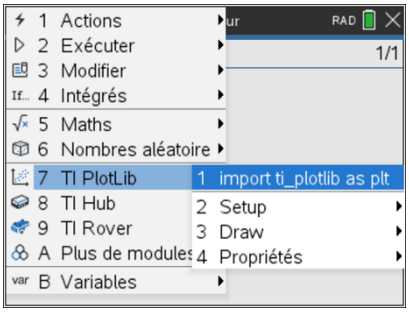
Dans cette seconde leçon de l'unité 4, vous allez découvrir comment représenter graphiquement une fonction en utilisant la librairie Python **Ti PlotLib**.

Objectifs :

- Représenter graphiquement une fonction.
- Réinvestir la notion de boucle fermée FOR.
- Paramétrer la représentation graphique.

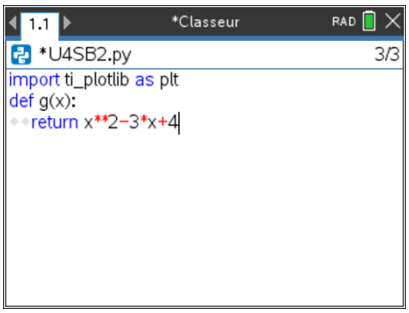
On se propose de réaliser un script utilisant la librairie **Ti PlotLib** et permettant de tracer la représentation graphique d'une fonction pour x prenant ses valeurs dans un intervalle $[a\,;\,b]$ avec N segments.

Le script que vous allez créer sera très général afin de pouvoir être réinvesti pour d'autres exemples.



Conseils à l'enseignant : Si la notion de boucle ne vous est pas familière, vous êtes invités à travailler les unités 1,2 et 3 du TI-Code Python.

- Commencer un nouveau script et le nommer U4SB2.
- Importer le module **TI PlotLib**, celui-ci s'obtient en appuyant sur la touche menu puis en choisissant **7 : TI PlotLib.**
- Définir la fonction $g : x \longmapsto x^2 - 3x + 4$.



Conseils à l'enseignant : la liste des abscisses est construite à l'aide d'une boucle fermée dont l'instruction se trouve en appuyant sur menu puis **4 Intégrés** et enfin **2 Contrôle** enfin en choisissant dans les instructions de contrôle **4 : for index in range(size) :**
En revanche celle des ordonnées **ly** est construite à partir de la liste **lx** des abscisses. On choisira donc l'instruction **7 : for index in list**.

Unité 4 : Utiliser la librairie ti_plotlib — Compétence 2 : Représenter graphiquement une fonction

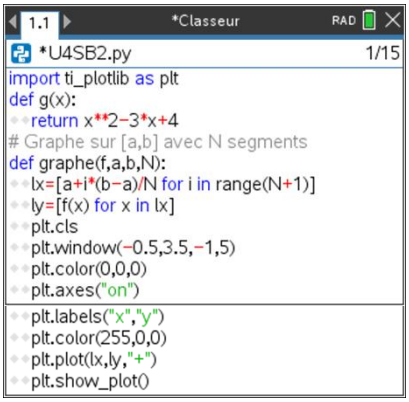
Dans cette seconde leçon de l'unité 4, vous allez découvrir comment représenter graphiquement une fonction en utilisant la librairie Python **Ti PlotLib**.

Objectifs :

- Représenter graphiquement une fonction.
- Réinvestir la notion de boucle fermée FOR.
- Paramétrer la représentation graphique.

A présent : vous êtes prêts pour effectuer la représentation graphique de la fonction. Insérer les instructions permettant de :

- Nettoyer l'écran : **plt.cls( )**.
- Régler les paramètres de la fenêtre graphique : **plt.window(xmin, xmax, ymin, ymax)**.
- Afficher les axes du repère : **plt.axes(« on »)**.
- Afficher le nom des axes : **plt.labels(« x », « y »)**.
- Effectuer la représentation graphique : **plt.plot(lx,ly, « + »)**.
- Afficher la représentation graphique : **plt.show_plot( )**.

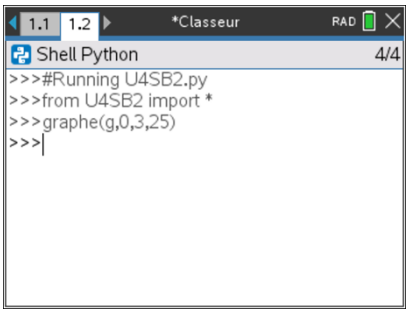


L'ensemble de ces instructions se trouve dans le module **TI PlotLib,** les paramètres de réglages de la représentation graphique dans le menu **Setup** et l'instruction de représentation graphique dans le menu Dessin puis **5 plot(x-list,y-list, « mark »)**.

Remarque : les instructions pour le choix de la couleur en RVB (rouge, vert, bleu) sont à coder entre 0 et 255 et à placer judicieusement, afin d'éviter par exemple d'avoir les axes en rouge.

Exécuter le script puis choisir la fonction **graphe( )** en appuyant sur la touche [var].

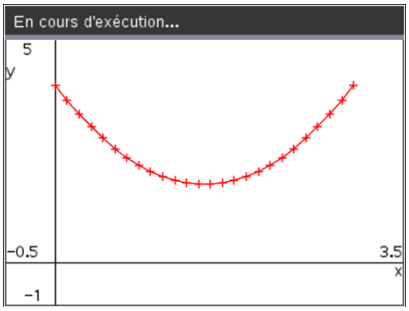
Demander la représentation graphique de la fonction contrainte sur l'intervalle [0 ; 3] avec 25 segments.



Conseils à l'enseignant : Si vous souhaitez un grand nombre de segments, préférez une représentation avec des pixels (**.** plutôt que **+**).

Prolongements possibles :

- Afficher un quadrillage.
- Changer de fonction ou en étudier plusieurs.
- Réaliser une étude mathématique (calcul différentiel) ; représenter par exemple la fonction avec 6 segments puis 50.



Unité 4 : Utiliser la librairie ti_plotlib — Compétence 2 : Représenter graphiquement une fonction

Dans cette troisième leçon de l'unité 4, vous allez découvrir comment représenter graphiquement une série de données, puis rechercher un modèle mathématique s'ajustant au mieux au nuage de points.

Objectifs :

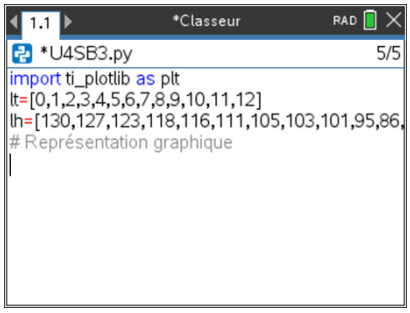
- Réaliser la représentation d'un nuage de points.
- Rechercher et utiliser un modèle de régression.
- Utiliser les listes en Python.

Position du problème : Lors d'une inondation, afin de fournir à la population des informations pratiques, les autorités enregistrent, chaque heure à partir du début de la décrue, la hauteur d'eau maximale par rapport à un point de référence.
Les données sont fournies dans le tableau ci-dessous. On souhaite représenter le nuage de points en utilisant la bibliothèque **TI PlotLib**, puis rechercher un modèle mathématique permettant de faire une extrapolation, afin de prévoir le temps total de la décrue.

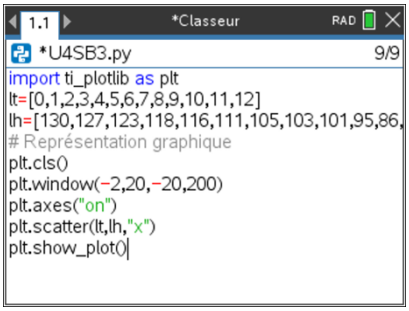
T(h)	0	1	2	3	4	5	6	7	8	9	10	11	12
H(cm)	130	127	123	118	116	111	105	103	101	95	86	80	71

Mise en œuvre : Commencer un nouveau script et le nomme U4SB3.

1) Importer le module **TI PlotLib**.
2) Enregistrer les données dans deux listes « **lt** » et « **lh** ».

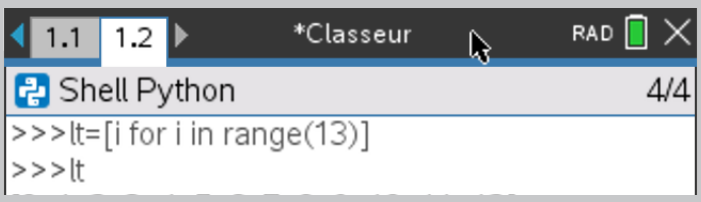


3) Préparer ensuite la représentation graphique :
   - Effacer l'écran **plt.cls( )**.
   - Régler les paramètres de la fenêtre graphique : x_{min} = -2 ; x_{max} = 20 ; y_{min} = -20 ; y_{max} = 200 en écrivant **plt.window(**-2, 20, -20, 200**)**.
   - Afficher les axes **plt.axes( )**.
   - Afficher la représentation graphique sous forme d'un nuage de points **plt.scatter()**.



Les instructions de paramétrage de la fenêtre graphique se trouvent dans le module **TI PlotLib** dans le menu **Setup**. La syntaxe de la représentation du nuage de points se situe quant à elle dans le menu **Draw.**

Conseil à l'enseignant : la liste des données du temps, en heures, peut être complétée à l'aide d'une boucle fermée.

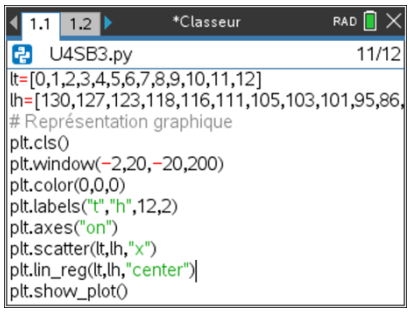


Dans cette troisième leçon de l'unité 4, vous allez découvrir comment représenter graphiquement une série de données, puis rechercher un modèle mathématique s'ajustant au mieux au nuage de points.

Objectifs :

- Réaliser la représentation d'un nuage de points.
- Rechercher et utiliser un modèle de régression.
- Utiliser les listes en Python.

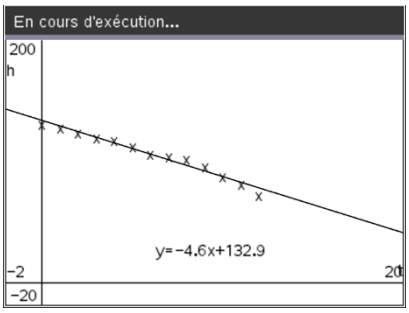
Exécuter le script et observer la représentation graphique. Si tout se passe correctement, vous devriez obtenir l'écran ci-contre.



L'ensemble des points semble-t-il aligné ?
Vous allez maintenant rechercher le modèle mathématique (affine) passant au mieux par l'ensemble des points du nuage.

Pour cela, vous devez rajouter la ligne **plt.lin_reg(xliste,yliste, « center »)** dans votre script afin que la droite de régression soit calculée à partir des listes de données **lt** et **lh**, puis affichée et centrée sur l'écran.

Afin de connaitre l'heure de la décrue totale, il vous reste à résoudre l'équation du premier degré $-4.64x + 132.90 = 0$.



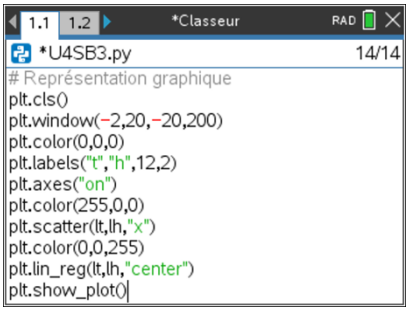
Unité 4 : Utiliser la librairie ti_plotlib — Compétence 2 : Représenter graphiquement une fonction

Dans cette troisième leçon de l'unité 4, vous allez découvrir comment représenter graphiquement une série de données, puis rechercher un modèle mathématique s'ajustant au mieux au nuage de points.

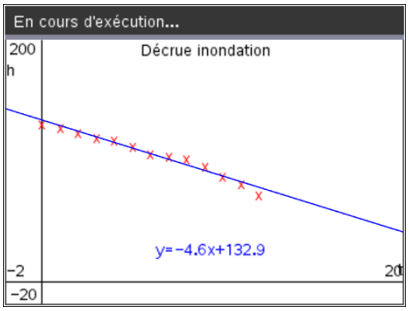
Objectifs :

- Réaliser la représentation d'un nuage de points.
- Rechercher et utiliser un modèle de régression.
- Utiliser les listes en Python.

Améliorons notre représentation graphique :
Inclure dans votre script des instructions permettant de conserver les axes en noir plt.color(0,0,0), puis une représentation graphique du nuage de point en rouge plt.color(255,0,0), et enfin une représentation en bleu de la droite de régression plt.color(0,0,255).



Il est aussi intéressant de rajouter un titre et des étiquettes aux axes.



Conseils à l'enseignant : Afin d'éviter des problèmes de superposition pour l'affichage, il convient de choisir des noms courts pour les étiquettes des axes.
L'étiquette des axes « t » et « h » peut être configurée selon vos souhaits en utilisant l'instruction **labels(« x-étiq », « y-étiq » »,x, y)**, **x** et **y** représentant la ligne où sont écrits ces labels. **Par défaut ces lignes sont respectivement 12 et 2 pour x et y et respectivement justifiées à gauche et à droite.**
De même, l'équation de régression peut être affichée à l'emplacement souhaité, en utilisant la commande **lin_reg(xlist, ylist, « disp », row)**. Par défaut, cette équation est affichée à la ligne 11.

Unité 4 : Utiliser la librairie ti_plotlib

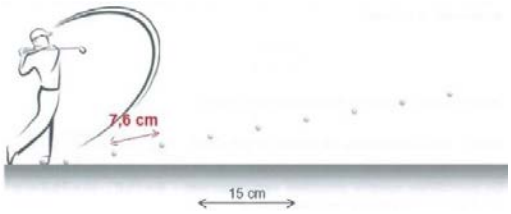
Application : Positions successive d'un mouvement

Dans l'application de l'unité 4, vous allez découvrir comment étudier un mouvement à partir des mesures effectuées sur une chronophotographie et réinvestir les connaissances acquises lors de l'utilisation du module **TIPlotLib**.

Objectifs :

- Représenter un nuage de points.
- Calculer puis représenter les vecteurs d'un système modélisé par un point.

Le problème : On étudie le mouvement d'une balle de golf à partir d'une chronophotographie (enregistrement d'un mouvement par prise de photographies selon un intervalle de temps fixé). On souhaite représenter l'évolution du vecteur vitesse au cours du temps.
Les positions sont relevées toutes les 0,066 s et sont rassemblées dans un tableau.



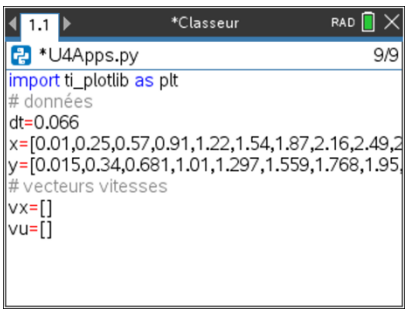
Remarque : dans l'unité 5, vous apprendrez à importer les mesures depuis les listes de la calculatrice en utilisant le module TI System.

t(s)	0	0.066	0.132	0.198	0.264	0.33	0.396	0.462	0.528	0.594	0.66
x(m)	0.01	0.25	0.57	0.91	1.22	1.54	1.87	2.16	2.49	2.81	3.15
y(m)	0.015	0.34	0.681	1.01	1.297	1.559	1.768	1.95	2.08	2.158	2.193

Mise en œuvre :

1 : Entrée des mesures et création des valeurs du temps.

- Commencer un nouveau script et le nommer U4APPS.
- Importer la bibliohèque **TI PlotLib** de représentation graphique.
- Entrer les mesures correspondant aux coordonnées d'un point repéré lors de l'analyse de la chronophotographie.

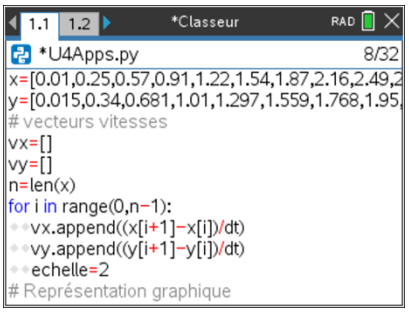


2 : Calcul des vecteurs vitesses.

Le vecteur vitesse à un instant t est donné par la relation : $\vec{V_i} = \dfrac{\overrightarrow{M_i M_{i+1}}}{t_{i+1} - t_i}$

On ne peut donc pas dessiner le vecteur vitesse du dernier point de la liste, il faut en tenir compte dans le code.

Astuce : Afin de rendre la représentation graphique lisible, on utilisera un facteur d'échelle de 2.



Unité 4 : Utiliser la librairie ti_plotlib

Application : Positions successive d'un mouvement

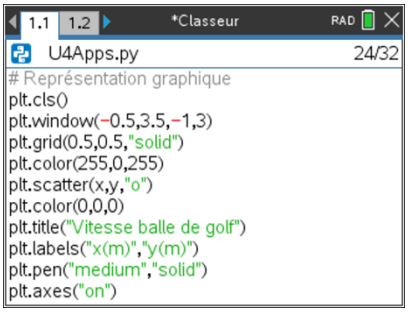
Dans l'application de l'unité 4, vous allez découvrir comment étudier un mouvement à partir des mesures effectuées sur une chronophotographie et réinvestir les connaissances acquises lors de l'utilisation du module **TIPlotLib**.

Objectifs :

- Représenter un nuage de points.
- Calculer puis représenter les vecteurs d'un système modélisé par un point.

3 : Régler les paramètres de la représentation graphique :

- **plt.cls( )** pour effacer l'écran.
- **plt.title(« titre »)** pour donner un titre à la représentation graphique.
- **plt.window(x_{min}, x_{max}, y_{min}, y_{max})** pour régler la fenêtre graphique.
- **plt.grid(xscl, ysc, « type »)** pour afficher une grille de graduation 0.5.
- **plt.color(255,0,255)** pour un affichage en couleur magenta.
- **plt.scatter(xlist, ylist, « type »)** pour un affichage sous forme d'un nuage de points.
- **plt.color(0,0,0)** pour un affichage en couleur noire pour les axes.
- **plt.pen(« medium », « solid »)** pour un affichage des axes en épaisseur moyenne.
- **plt.labels(« x(m) », « y(m) »)** pour afficher les étiquettes aux lignes 12 et 2 par défaut.

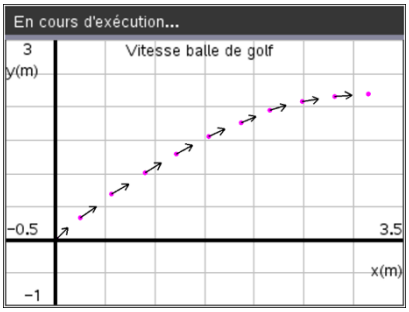


Le tracé des vecteurs est effectué au sein d'une boucle fermée de *n*-1 valeurs.

- **plt.line(x0,y0,x1,y1)** pour le tracé d'un vecteur vitesse.
- **plt.show( )** pour afficher la représentation graphique.

Rappel : La position de la balle entre les instants t_i et t_{i+1} peut-être repérée dans un repère cartésien par $x_i + vx_i \times dt$ pour l'abscisse et $y_i + vy_i \times dt$ pour l'ordonnée.

Exécuter votre script ; vous devriez obtenir une représentation graphique analogue à celle de l'écran ci-contre.



Unité 4 : Utiliser la librairie ti_plotlib

Application : Positions successive d'un mouvement

Dans l'application de l'unité 4, vous allez découvrir comment étudier un mouvement à partir des mesures effectuées sur une chronophotographie et réinvestir les connaissances acquises lors de l'utilisation du module **TIPlotLib**.

Objectifs :

- Représenter un nuage de points.
- Calculer puis représenter les vecteurs d'un système modélisé par un point.

Conseil à l'enseignant : En appuyant sur la touche [tab], vous pouvez facilement compléter les différents champs de chaque instruction.
Utilisez les touches [ctrl][C] ; [ctrl][V] afin de copier-coller une instruction.

Pour des scripts complexes, vous disposez également de l'opportunité de dupliquer un script. Pour cela, afficher la liste des scripts, placer le curseur devant le nom du script à dupliquer, puis appuyer sur [menu] puis **1 Action** et choisir le menu **3 : Créer une copie**. Un nouveau nom vous sera demandé.

Si le nombre de données est important ou provient d'une expérience réalisée avec une console d'acquisition, il est possible d'utiliser l'application « Tableur et listes » afin de copier facilement des données pour les sauvegarder sous forme de listes avant de les importer (voir Unité 5 Compétence 1).

Unité 5 : Utiliser la bibliothèque TI System

Compétence 1 : Travailler sur des données

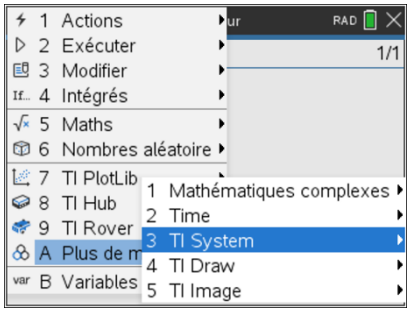
Dans cette première leçon de l'unité 2, vous allez découvrir comment utiliser la bibliothèque **Ti System** pour importer ou exporter des listes dans un script Python.

Objectifs :

- Importer-exporter des listes.
- Réinvestir les notions de l'unité 4 sur les représentations graphiques.

La bibliothèque ou module T**i System** utilisée seule ou en complément des autres, permet de communiquer (dans les deux sens) avec la calculatrice graphique.

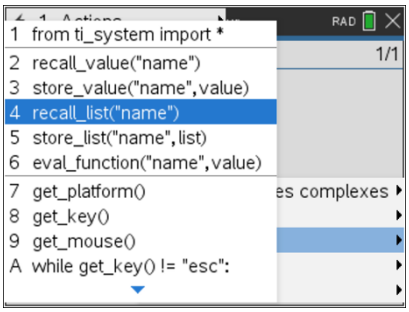
Pour charger cette bibliothèque, appuyer sur la touche [menu] puis **A Plus de modules** et enfin **3 TI System**.



Dans cette leçon, nous allons concentrer notre attention sur l'utilisation des instructions :

4 : var=recall_list(« nom ») et 5 : store_list(« nom »,var)

Les autres options de cette bibliothèque seront abordées dans les autres leçons de cette unité.

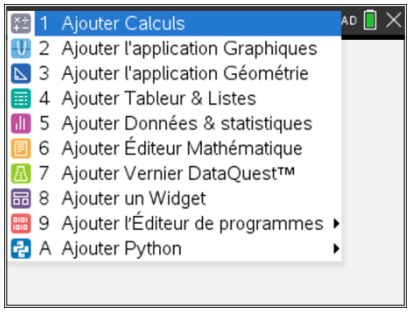


1 : Importer des données depuis la calculatrice.

a) Création de deux listes.

Dans un premier temps nous allons simplement créer deux listes contenant les données.

- Créer une nouvelle page de calculs [ctrl] [I] puis choisir **1 Ajouter Calculs.**
- Le même travail est réalisable à partir de l'application **Tableur&Listes**.



Unité 5 : Utiliser la bibliothèque TI System

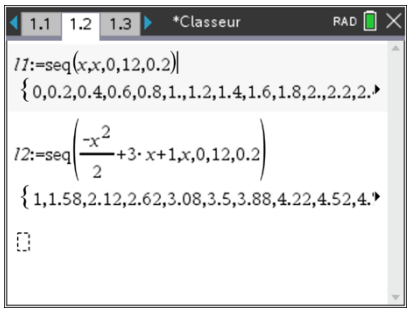
Compétence 1 : Travailler sur des données

Dans cette première leçon de l'unité 2, vous allez découvrir comment utiliser la bibliothèque **Ti System** pour importer ou exporter des listes dans un script Python.

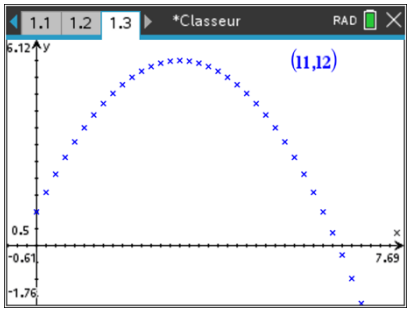
Objectifs :

- Importer-exporter des listes.
- Réinvestir les notions de l'unité 4 sur les représentations graphiques.

Vous allez créer dans la liste $\mathbf{l_1}$ une suite de nombre entre 0 et 12 avec un pas de 0.2 .
Puis une liste $\mathbf{l_2}$ correspondant à la liste des images de x (parcourant la liste $\mathbf{l_1}$ par la fonction f définie par : $x \longmapsto -x^2/2 + 3x + 1$



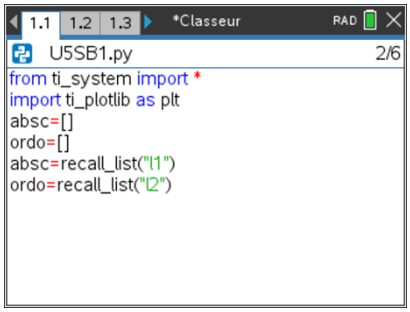
- Insérer une nouvelle page « **Application graphique** ».
- Représenter graphiquement le nuage de points ($\mathbf{l_1}$,$\mathbf{l_2}$).



- Régler les paramètres de la fenêtre graphique tels que : X_{min} = -1.2 ; X_{max} = 8 ; Y_{min}= -0.5 et enfin Y_{max}= 6.5.

b) Import des données dans un script Python.

- Commencer un nouveau script et le nommer U5SB1.
- A partir de menu importer la bibliothèque **TI System**.
- Créer deux variables listes (vides) **absc** et **ordo**.
- Importer les bibliothèques **TI System** et **TI PlotLib** (il n'y a pas d'ordre particulier à respecter).
- Créer une variable **absc** puis, à partir des options de la bibliothèque **TI System**, choisir l'option **4 : var=recall_list(« nom »)**. Comme les abscisses sont dans la liste $\mathbf{l_1}$, le champ « nom » est complété par le nom complet de la liste.
- Créer une autre variable **ordo** et procéder de la même manière avec la liste $\mathbf{l_2}$.



Unité 5 : Utiliser la bibliothèque TI System

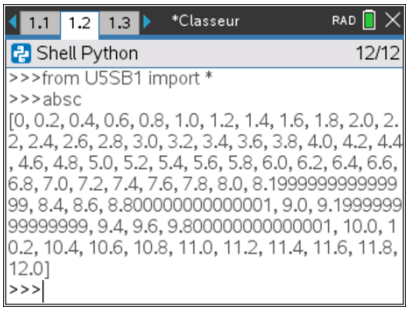
Compétence 1 : Travailler sur des données

Dans cette première leçon de l'unité 2, vous allez découvrir comment utiliser la bibliothèque **Ti System** pour importer ou exporter des listes dans un script Python.

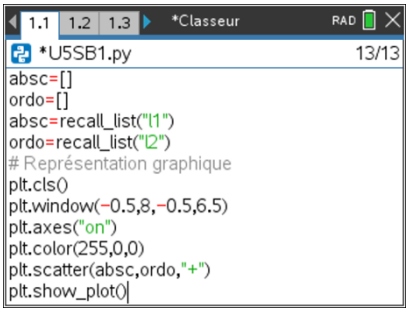
Objectifs :

- Importer-exporter des listes.
- Réinvestir les notions de l'unité 4 sur les représentations graphiques.

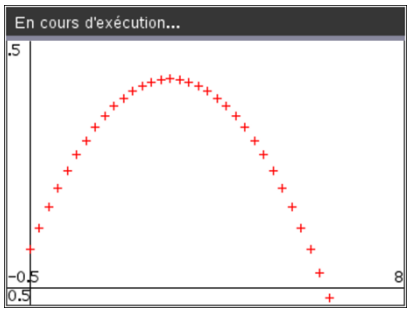
- Exécuter le script, puis vérifier le contenu de vos variables **absc** et **ordo** en appuyant sur la touche [var].
- Rappeler ensuite le nom de la « liste » **absc**, puis valider [enter].
- Procéder de la même façon avec la liste des ordonnées (**ordo**).



c) Représentation graphique.
Paramétrer votre représentation graphique comme proposé sur l'écran ci-contre.



Exécuter votre script [ctrl] [R].

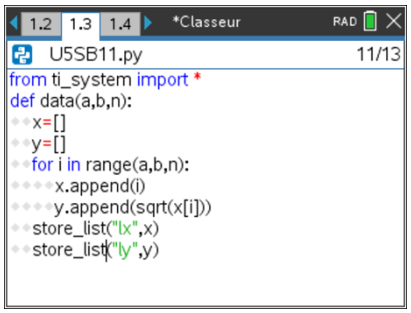


2 : Exporter des données

Créer un nouveau script et le nommer U5SB11.

Conseil à l'enseignant : Placer le curseur à la fin d'une ligne et valider. L'ordre d'écriture de l'importation des modules est sans importance.

- Créer une fonction nommée **data(a,b,n)**.
- Vous allez créer deux listes de données, à représenter sous forme d'un nuage de points, comportant des valeurs dans un intervalle **[a ; b],** calculées avec un pas **n**.
- Dans la liste **y**, on calcule la racine carrée des valeurs de la liste **x**.
- Pour créer ces listes de données, nous allons construire une boucle fermée après avoir bien entendu créé deux listes vides.



Unité 5 : Utiliser la bibliothèque TI System

Compétence 1 : Travailler sur des données

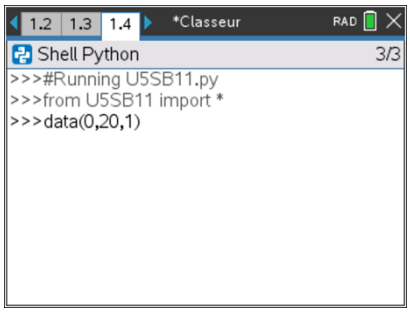
Dans cette première leçon de l'unité 2, vous allez découvrir comment utiliser la bibliothèque **Ti System** pour importer ou exporter des listes dans un script Python.

Objectifs :

- Importer-exporter des listes.
- Réinvestir les notions de l'unité 4 sur les représentations graphiques.

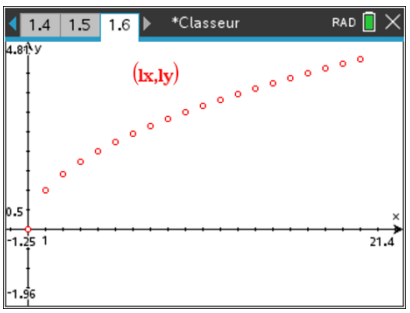
Conseil à l'enseignant : La création de deux listes vides évite le renvoi d'un message d'erreur lors de l'exécution du script.

Remarque : Attention à l'indentation, les instructions **store_list** n'ont pas à être dans la boucle. Utiliser [del] afin de supprimer un niveau de décalage.



- Exécuter votre script. On prend ici 21 valeurs de 0 à 20 par pas de 1
- Sortir de l'environnement Python et afficher la représentation graphique de vos listes **lx** et **ly** (nuage de points dans l'application graphique par exemple).

Conseil à l'enseignant : l'export vers les listes de la calculatrice sera particulièrement intéressant pour représenter des données acquises par l'intermédiaire de capteurs avec le Microcontrôleur **TI-Innovator™** & **Robot TI-Innovator™ Hub**.



Unité 5 : Utiliser la bibliothèque TI System

Compétence 2 : Modélisation

Dans cette seconde leçon de l'unité 5, vous allez découvrir comment importer les résultats d'une modélisation en utilisant la bibliothèque **TI System**.

Objectifs :

- Effectuer une modélisation linéaire.
- Importer les résultats de cette modélisation dans un script Python.

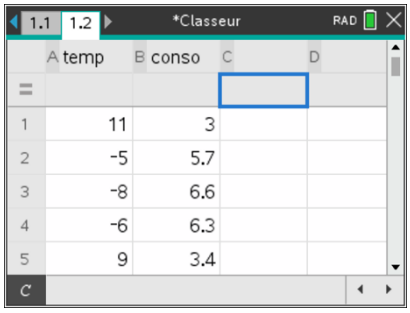
Vous allez dans cette leçon effectuer une modélisation linéaire à partir de données préalablement inscrites dans les listes de la calculatrice. Ensuite, vous écrirez un script afin d'importer les résultats de cette modélisation afin de les utiliser pour une représentation graphique, une interpolation ou extrapolation…

Le problème : Dans une journée, le pic de consommation d'électricité est atteint vers 19 h. Au niveau national, on a enregistré un pic de consommation de 96 350 mégawatts le mercredi 15 décembre 2019 à 19h02. Vous désirez prévoir les pics de consommation du prochain week-end pour la zone directement raccordée à la centrale.

Pour établir cette prévision, vous disposez de dix relevés de consommations réalisés à 19h en fonction de la température et présentés dans le tableau ci-dessous.

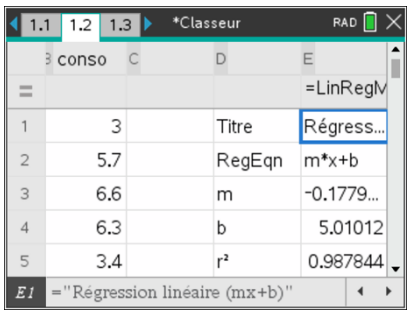
T°C	11	-5	-8	-6	9	14	4	-1	-12	3
MW	3	5.7	6.6	6.3	3.4	2.7	4	5.1	7.1	4.6

- Les données sont entrées dans les listes de la calculatrice, la température dans **temp** et la consommation en MW dans **conso**.
- Pour cela, ouvrir une application **tableur &listes**, puis entrer les données.



Remarque : Il est également possible d'utiliser l'application **Données & Statistiques**, mais aussi de définir les listes à partir de l'application **Calculs**.

- Effectuer une régression linéaire sous la forme mx+b (menu puis **4 Statistiques**).



Unité 5 : Utiliser la bibliothèque TI System

Compétence 2 : Modélisation

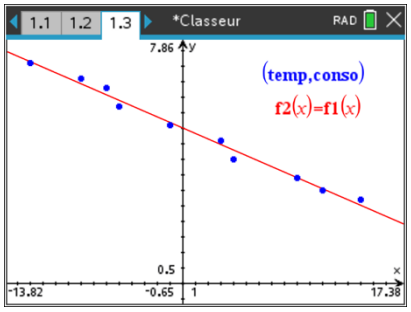
Dans cette seconde leçon de l'unité 5, vous allez découvrir comment importer les résultats d'une modélisation en utilisant la bibliothèque **TI System**.

Objectifs :

- Effectuer une modélisation linéaire.
- Importer les résultats de cette modélisation dans un script Python.

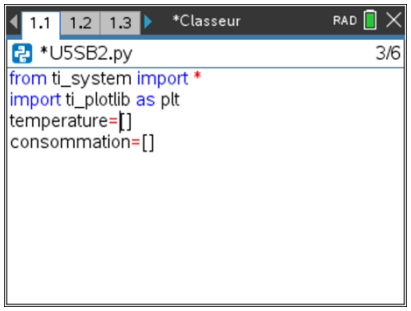
La fonction permettant de prévoir la consommation en fonction de la température est donc : $C = -0.18 \times t + 5.01$.

Représenter dans l'application **Graphiques**, le nuage de points ainsi que la droite de régression.

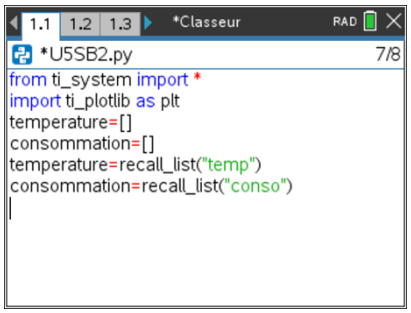


Utilisation des résultats de la modélisation dans un script Python.

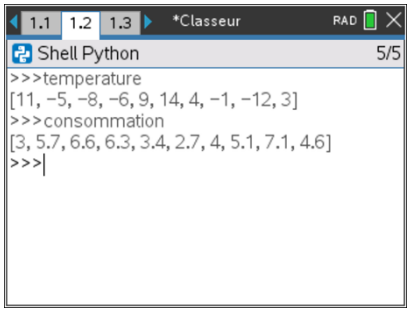
- Commencer un nouveau script et le nommer U5SB2.
- Incorporer le menu **TI System** et **TI Plotlib**.
- Créer deux listes **temperature** et **consommation** vides.



Rappeler le contenu des listes **temp** et **conso** dans leurs noms respectifs. L'instruction **var=recall_list(« nom »)** est accessible dans le menu du module **TI System**(voir Unité 5 Compétence 1).



Tester votre script et demander l'affichage des différentes variables ainsi créées en appuyant sur la touche var, puis en choisissant vos variables.



Unité 5 : Utiliser la bibliothèque TI System

Compétence 2 : Modélisation

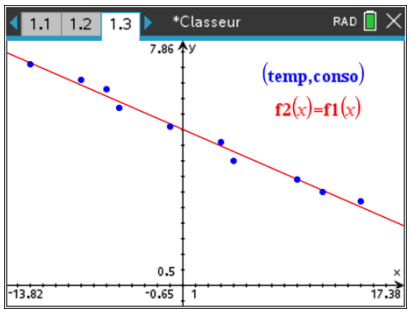
Dans cette seconde leçon de l'unité 5, vous allez découvrir comment importer les résultats d'une modélisation en utilisant la bibliothèque **TI System**.

Objectifs :

- Effectuer une modélisation linéaire.
- Importer les résultats de cette modélisation dans un script Python.

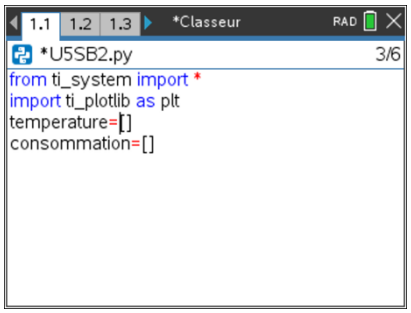
La fonction permettant de prévoir la consommation en fonction de la température est donc : $C = -0.18 \times t + 5.01$.

Représenter dans l'application **Graphiques**, le nuage de points ainsi que la droite de régression.

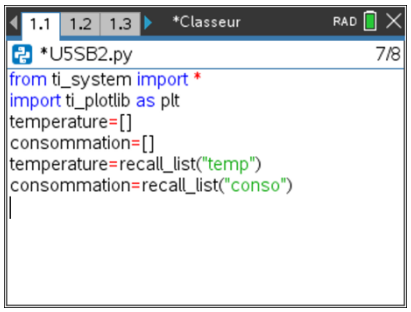


Utilisation des résultats de la modélisation dans un script Python.

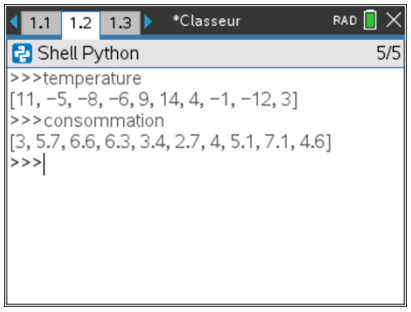
- Commencer un nouveau script et le nommer U5SB2.
- Incorporer le menu **TI System** et **TI Plotlib**.
- Créer deux listes **temperature** et **consommation** vides.



Rappeler le contenu des listes **temp** et **conso** dans leurs noms respectifs. L'instruction **var=recall_list(« nom »)** est accessible dans le menu du module **TI System**(voir Unité 5 Compétence 1).



Tester votre script et demander l'affichage des différentes variables ainsi créées en appuyant sur la touche var, puis en choisissant vos variables.



Unité 5 : Utiliser la bibliothèque TI System

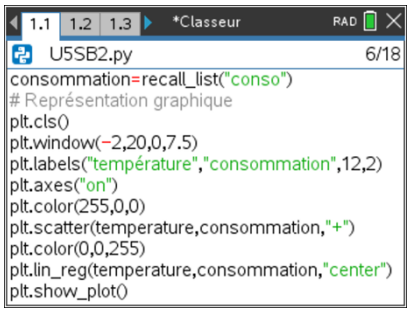
Compétence 2 : Modélisation

Dans cette seconde leçon de l'unité 5, vous allez découvrir comment importer les résultats d'une modélisation en utilisant la bibliothèque **TI System**.

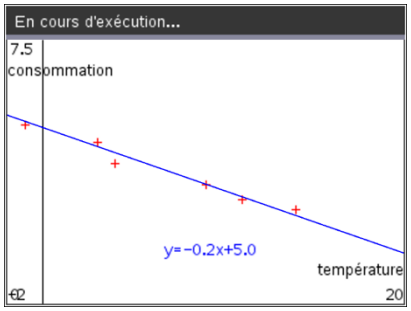
Objectifs :

- Effectuer une modélisation linéaire.
- Importer les résultats de cette modélisation dans un script Python.

Pour compléter cette leçon et réinvestir les compétences acquises lors de l'unité 4, vous pouvez effectuer la représentation graphique de vos mesures ainsi que du modèle de régression calculé.



Conseil à l'enseignant : l'instruction **lin_reg(xliste, yliste, « aff », row)** comporte un paramètre supplémentaire **row** donnant la possibilité de placer sur une autre ligne l'équation de régression. Par défaut, celle-ci est placée à la ligne 11.



Unité 5 : Utiliser la bibliothèque TI System

Compétence 2 : Modélisation

Dans cette seconde leçon de l'unité 5, vous allez découvrir comment importer les résultats d'une modélisation en utilisant la bibliothèque **TI System**.

Objectifs :

- Effectuer une modélisation linéaire.
- Importer les résultats de cette modélisation dans un script Python.

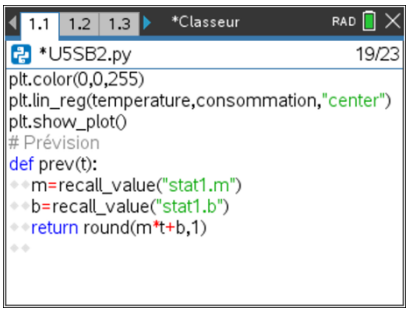
Prolongement : prévoir la consommation pour une température donnée.
Vous allez récupérer dans deux variables **a** et **b** les coefficients de l'équation afin de les utiliser dans une fonction vous permettant de réaliser ainsi une extrapolation ou une interpolation de la consommation électrique, lorsque la température est connue.

Définir une fonction **prev(t)** qui retournera la consommation prévue pour une température **t**.

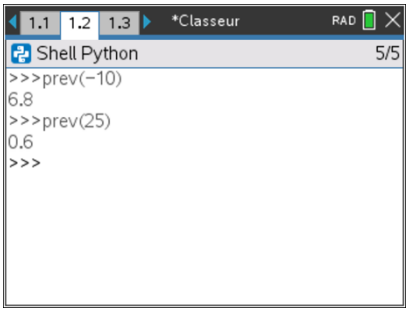
Exécuter votre script et effectuer quelques tests pour différentes températures.
On rappelle que le modèle représente une prévision de consommation électrique à 19h02 en fonction de la température.

Vous pouvez ainsi déterminer les limites de validité de votre modèle.

- L'instruction **recall(value , « name »)** permet de rappeler au sein d'un script Python une variable définie ou déclarée au sein d'une application.
- Attention : les variables statistiques sont précédées du nom **stat n°.nom.** Pour les retouver, appuyer sur la touche [var] lorsque l'application est active.



Exemple :
Prévoir la consommation électrique en MW pour une température de -10°C puis de 25°C.



Unité 5 : Utiliser la librairie ti_system

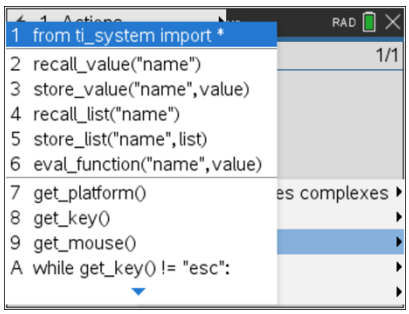
Compétence 3 : Affichage et temporisation

Dans cette troisième leçon de l'unité 5, vous allez découvrir comment utiliser les options d'affichage et de « temporisation » de la librairie Python **TI System**.

Objectifs :

- Comprendre le fonctionnement des instructions **disp…**
- Utiliser ces instructions dans un script en complément de celles des autres modules.

Lors des leçons 1 et 2 de cette unité, vous avez appris à importer-exporter des listes de données puis à travailler sur une équation de régression.
La librairie TI System comporte d'autres options parfois utiles que l'on se propose, que l'on se propose de découvrir ici.

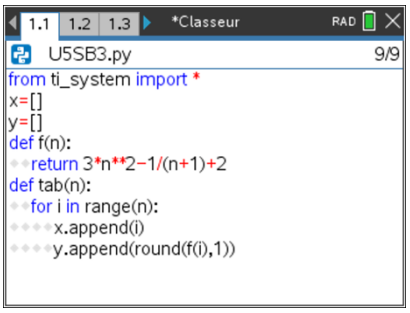


1 : L'instruction clear_history

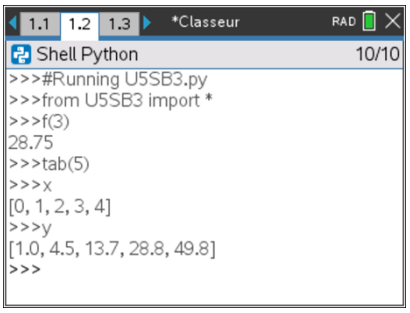
- Commencer un nouveau script et le nommer U5SB3.
- Importer le module **TI System**.

- Écrire un script permettant de tabuler une fonction.

$$x \to 3x^2 - \frac{1}{x+1} + 2$$



- Rappeler votre script U5SB3 et demander son exécution. Vous devriez obtenir l'écran ci-contre.



Unité 5 : Utiliser la librairie ti_system

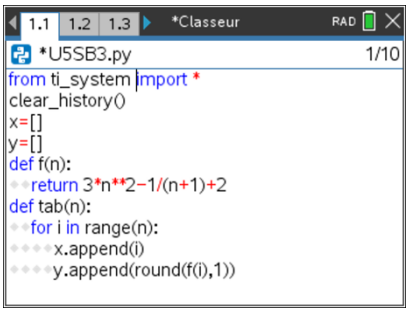
Compétence 3 : Affichage et temporisation

Dans cette troisième leçon de l'unité 5, vous allez découvrir comment utiliser les options d'affichage et de « temporisation » de la librairie Python **TI System**.

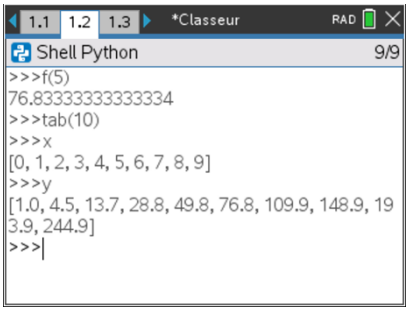
Objectifs :

- Comprendre le fonctionnement des instructions **disp…**
- Utiliser ces instructions dans un script en complément de celles des autres modules.

L'instruction **clear_history** nettoie l'écran et place le prompt de la console en haut de l'écran. Vous obtenez ainsi au sein d'un programme un écran débarrassé de ses précédents affichages.



Ainsi l'exécution du script est rendue plus lisible.

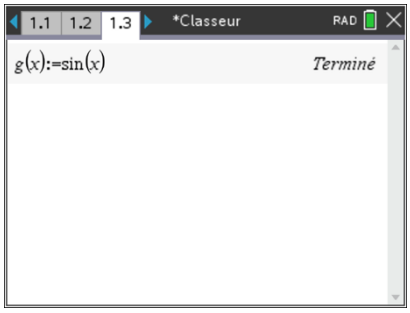


2 : L'instruction eval_function()

- Insérer une page de calculs [ctrl] [I] et définir la fonction :

$$g: x \to sin(x)$$

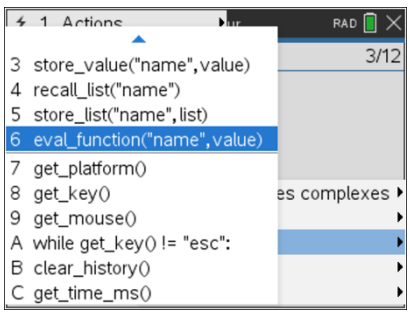
- L'instruction **eval_function()** va permettre d'utiliser dans un script Python, une instruction préalablement définie au sein d'une autre application de la TI-Nspire™ (Calculs, Graphiques, Tableur&Listes…).



Reprendre le script et le modifier comme sur l'écran ci-contre de façon à pouvoir tabuler la fonction g : puis de représenter graphiquement le nuage de points correspondant dans l'intervalle $[0\,;\,2\pi]$.

L'instruction **eval_function** se trouve dans le menu **TI System**, mais attention à sa syntaxe.

eval_function(« name »,value) : l'argument name sera ici **g** et non pas g(x)



Unité 5 : Utiliser la librairie ti_system

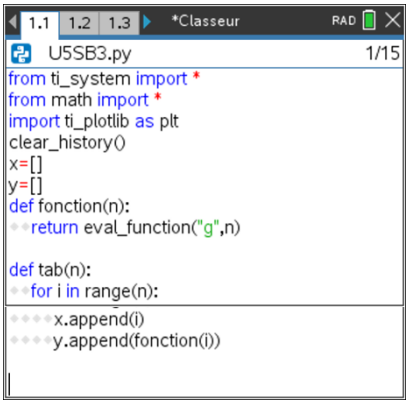
Compétence 3 : Affichage et temporisation

Dans cette troisième leçon de l'unité 5, vous allez découvrir comment utiliser les options d'affichage et de « temporisation » de la librairie Python **TI System**.

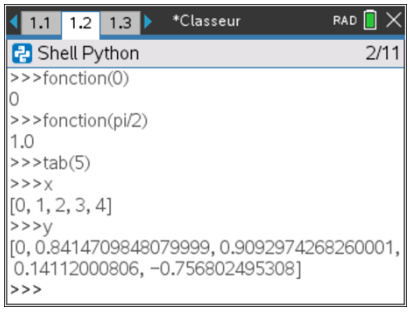
Objectifs :

- Comprendre le fonctionnement des instructions **disp…**
- Utiliser ces instructions dans un script en complément de celles des autres modules.

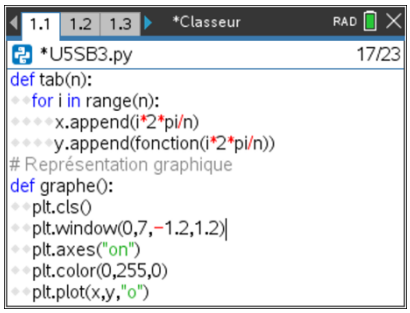
- Compléter en modifiant le script précédent.



- Exécuter le script. Attention, l'unité par défaut est le radian.
- Voir les options de la librairie Maths pour une expression des mesures des angles en degrés ou une conversion. (Unité 1, compétence 1).

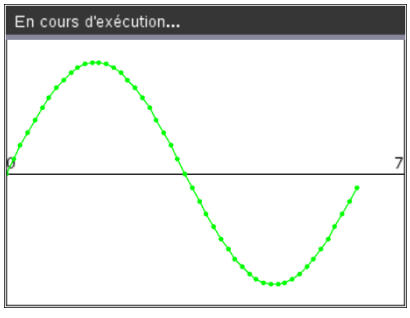


- Modifier à nouveau le script afin de calculer les coordonnées de n points dans l'intervalle $[0\,;\,2\pi]$. Vous pouvez réinvestir ici ce que vous avez appris à l'unité 4, compétence 2.



- Terminer le script en incluant une fonction **graphe( )**.

- Vérifier le fonctionnement du script.



Remarque : L'instruction **recall_value(« name »)** permettra de la même manière de réinvestir dans une variable d'un script Python, le contenu d'une variable utilisée dans une autre application de la TI-Nspire™.

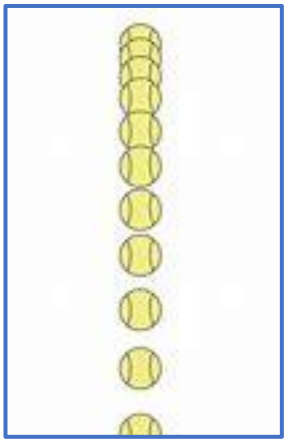
Unité 5 : Utiliser la librairie TI System & Ti_PlotLib

Application : Etude de la chute libre

Dans cette application de l'unité 5, vous allez réinvestir les notions vues lors dans les unités 4 et 5 afin de créer un simulateur permettant de décrire un mouvement.

Objectifs :

- Effectuer une simulation d'une chronophotographie.
- Exporter les résultats dans les listes de la calculatrice.



On propose dans cette application de l'unité 5, d'utiliser le langage Python afin de créer un simulateur d'un phénomène physique. Nous retenons l'étude de la chute libre :

- Le script devra permettre la description d'un mouvement ;
- La représentation graphique des données sous forme d'une chronophotographie ;
- L'export des données vers les listes de la calculatrice.

Le problème : Une balle est lâchée sans vitesse initiale d'une hauteur h. Les photographies sont effectuées au cours du temps toutes les 60 ms.

Vous allez écrire un script permettant de calculer, au cours du temps, la position de la balle, puis de la représenter graphiquement.

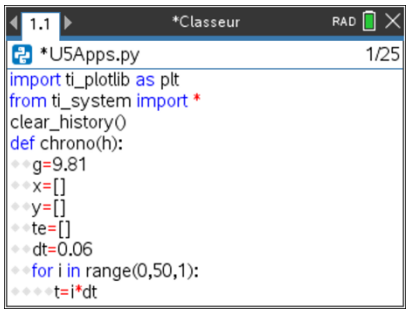
a) Calcul des positions successives

Lors de la chute libre, à partir d'une origine des temps et des espaces choisis (y>0), la position de la balle peut être calculée en utilisant la relation $y = -\frac{g}{2} \times t^2 + h_0$.

Rappelons que : $g \approx 9{,}81$ m.s^{-2}.

Créer un nouveau script et le nommer U5Apps.

- Importer les modules **TI System** et **TI PlotLib.**
- Nettoyer l'écran.
- Créer une fonction **chrono(h)** prenant comme paramètre la hauteur initiale à laquelle est lâchée la balle et qui affiche la position de la balle en fonction du temps.
- Créer trois listes vides, abscisse **x**, ordonnée **y** et temps **te.**
- Les ordonnées sont calculées toutes les 60 ms.
- Les valeurs sont sauvegardées dans les listes si y >0 (la balle ne s'enfonce pas dans le sol).



Unité 5 : Utiliser la librairie TI System & Ti_PlotLib

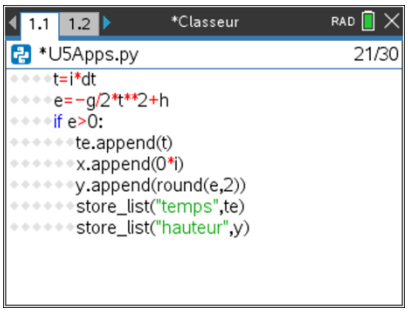
Application : Etude de la chute libre

Dans cette application de l'unité 5, vous allez réinvestir les notions vues lors dans les unités 4 et 5 afin de créer un simulateur permettant de décrire un mouvement.

Objectifs :

- Effectuer une simulation d'une chronophotographie.
- Exporter les résultats dans les listes de la calculatrice.

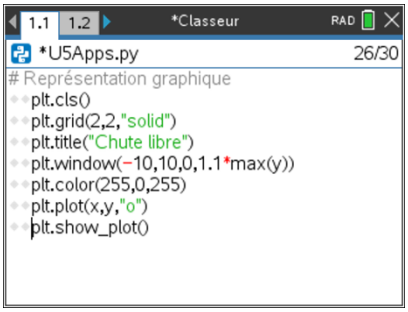
Créer une boucle fermée permettant de calculer l'altitude de la balle en fonction du temps. Les résultats sont exprimés à 10^{-2} près et stockés dans leurs listes respectives. Comme on souhaite avoir une représentation graphique de la chronophotographie correspondant à un cas réel, la liste des abscisses est complétée avec des 0.
Enfin, les listes **te** et **y** sont respectivement exportées dans les listes **temps** et **hauteur** de la calculatrice.



b) Représentation graphique

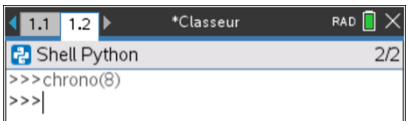
Paramétrer une représentation graphique :

- Nettoyer l'écran **plt.cls( )**.
- Afficher une grille d'unité 2 **plt.grid(xscl, yscl, type,(r,v,b))**.
- Régler la fenêtre graphique **plt.window(x_{min}, x_{max}, y_{min}, y_{max})**.
- Fixer la couleur du point au magenta **plt.color(255,0,255)**.
- Représenter le nuage de point **plt.plot(x-list, y-list, marque)**.
- Afficher le graphe **plt.show_plot( )**.



Exécuter votre script et appeler la fonction chrono [var], puis lui fournir comme argument la hauteur initiale du lâché (8m par exemple).

La chronophotographie est représentée.



Conseil à l'enseignant : A partir de la liste des positions de la balle, on pourra éventuellement calculer la vitesse de la balle, puis afficher les vecteurs vitesses.

Unité 5 : Utiliser la librairie TI System & Ti_PlotLib

Application : Etude de la chute libre

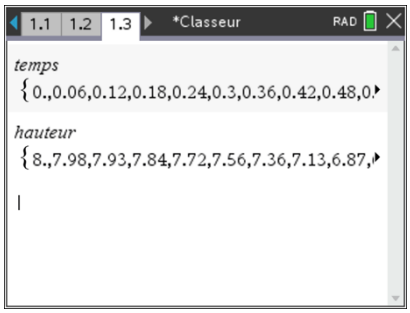
Dans cette application de l'unité 5, vous allez réinvestir les notions vues lors dans les unités 4 et 5 afin de créer un simulateur permettant de décrire un mouvement.

Objectifs :

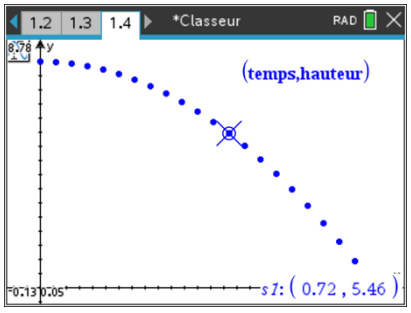
- Effectuer une simulation d'une chronophotographie.
- Exporter les résultats dans les listes de la calculatrice.

c) Visualisation des données exportées

- Quitter l'éditeur Python et afficher les listes.
- Insérer une application **Calculs**.
- Rappeler les listes **temps** et **hauteur**.



- Insérer une application **Graphiques** et représenter le nuage de points (**temps**, **hauteur**).
- Utiliser l'outil **Trace** afin d'explorer la représentation graphique.



Unité 6 : Utiliser les librairies TI Hub & TI Rover

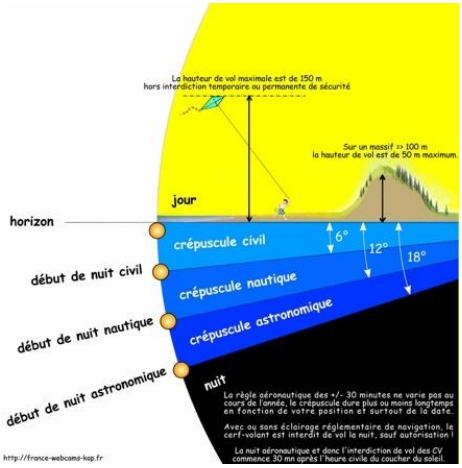
Compétence 1 : Les capteurs intégrés au hub

Dans cette première leçon de l'unité 6, vous allez découvrir comment utiliser la librairie **TI_Hub** afin de commander les dispositifs intégrés au hub TI-Innovator™.

Objectifs :

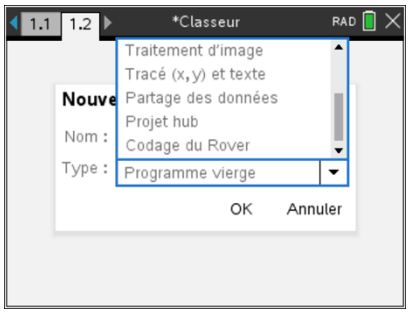
- Découvrir le module **TI-Hub**.
- Écrire un script intégrant la librairie TI Hub pour les dispositifs intégrés.

Vous allez, dans cette leçon, utiliser la librairie **TI Hub** afin de noter visuellement un changement de luminosité pour, par la suite, simuler un interrupteur crépusculaire, ou bien enregistrer une série de mesures lors du lever du soleil ou du crépuscule.
Vous commencerez par ailleurs à envisager comment associer cette librairie à celle que vous connaissez déjà (**TI PlotLib & TI System**) afin de créer un projet scientifique complet.
Le script que vous allez écrire correspond à l'algorithme simple suivant :




```
Mesurer l'intensité lumineuse ambiante :
Lum0 ← mesure ± (tolérance ?)
Modifier l'intensité lumineuse (lampe ; cache devant le capteur)
Lum1 ← mesure
  Si Lum1 > Lum 0 :
  |  Alors allumer la DEL RVB en rouge (2s)
  Sinon Si Lum0 < Lum1 :
  |  Alors allumer la DEL RVB en vert (2s)
  Sinon : Ne rien faire
```

Commencer un nouveau script et le nommer U6SB1
Ce script doit intégrer la librairie **TI Hub**. Pour cela, vous avez plusieurs possibilités.



a) Lors de la création d'un script Python, préciser le type général de script. Celui-ci intègrera à minima, la librairie correspondant au nom du type.
b) Vous pouvez également partir d'un script vierge, puis incorporer manuellement les librairies dont vous avez besoin.

Unité 6 : Utiliser les librairies TI Hub & TI Rover

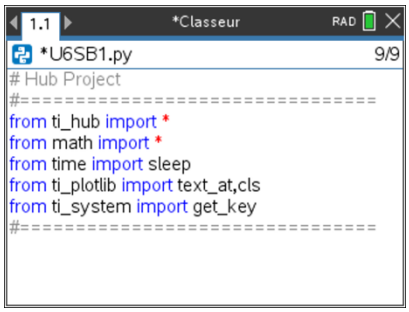
Compétence 1 : Les capteurs intégrés au hub

Dans cette première leçon de l'unité 6, vous allez découvrir comment utiliser la librairie **TI_Hub** afin de commander les dispositifs intégrés au hub TI-Innovator™.

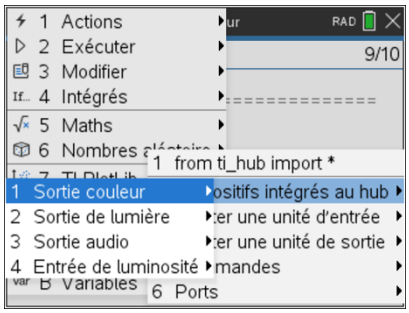
Objectifs :

- Découvrir le module **TI-Hub**.
- Écrire un script intégrant la librairie TI Hub pour les dispositifs intégrés.

En choisissant le type **Projet Hub**, Vous devriez obtenir l'écran ci-contre.

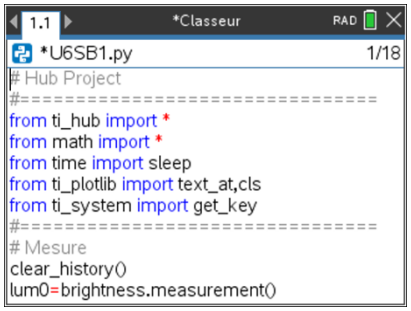


Nous allons utiliser le capteur de luminosité intégré au **TI-Innovator™**, ainsi que la diode RVB. Afin que le script soit en mesure de les gérer, intégrons les librairies correspondantes. Pour cela choisir dans le menu **Modul** puis **6 TI Hub…** et enfin dans le sous-menu **1 Dispositifs intégrés au hub**.

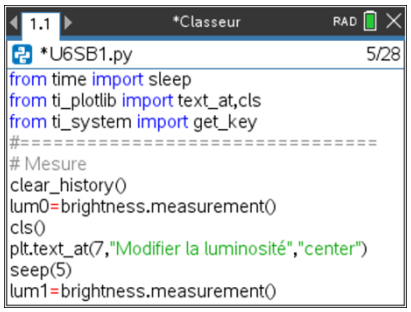


Conseil à l'enseignant : Les deux autres possibilités concerneront les capteurs et actionneurs que l'on connectera directement sur les ports d'entrée sortie IN… et OUT… du Hub ou éventuellement sur les ports BBx.

- Afin que le programme affiche des informations sur un écran « propre », nettoyer celui-ci, à l'aide l'instruction **clear_history()** se trouvant dans la librairie **TI System.**
- Créer une variable l**um0**, à laquelle on affecte une mesure de la luminosité. L'instruction **brightns.measurement()** se trouve dans le module **8 TI Hub ,** puis **2 Dispositifs intégrés au hub**.



- Afficher un message invitant l'utilisateur à modifier l'intensité lumineuse dans le voisinage du capteur intégré plt.text_at( ).
- Ajouter un délai sleep( ), le temps d'effectuer cette modification.
- Créer une variable lum1 à laquelle, on affecte la nouvelle mesure.



Unité 6 : Utiliser les librairies TI Hub & TI Rover

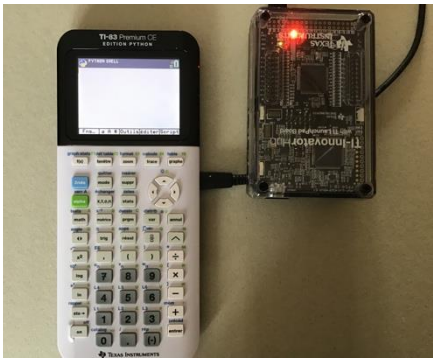
Compétence 1 : Les capteurs intégrés au hub

Dans cette première leçon de l'unité 6, vous allez découvrir comment utiliser la librairie **TI_Hub** afin de commander les dispositifs intégrés au hub TI-Innovator™.

Objectifs :

- Découvrir le module **TI-Hub**.
- Écrire un script intégrant la librairie TI Hub pour les dispositifs intégrés.

- Les mesures sont ensuite comparées. Selon le résultat de l'instruction conditionnelle, la DEL RVB s'allumera en vert ou rouge pendant un délai de 2 secondes.



```
1.1   *Classeur   RAD
*U6SB1.py   26/28
# Comparaison
if lum0<lum1:
    color.rgb(255,0,0)
    sleep(2)
    color.off()
elif lum0>lum1:
    color.rgb(0,255,0)
    sleep(2)
    color.off()
else:
    color.off()
```

Prolongement de l'activité :
Modifier le script précédent ou en créer un autre U6SB11
Ce nouveau script doit sur 40 minutes enregistrer les mesures de la luminosité.

Conseil à l'enseignant : Attention le capteur brightness du TI-Innovator™ n'est pas étalonné en Lux, mais cela n'a pas d'importance ici, dans la mesure où l'on ne s'intéresse qu'aux variations de la luminosité et non à leur mesure en Lux.

Les mesures sont sauvegardées dans une liste **r[ ]**.
Celles correspondant à la valeur du temps dans une liste **t[ ]**.

Le script proposé ci-dessous propose également la représentation graphique des mesures afin de le comparer à la représentation ci-contre.
Il est également proposé un export vers les listes de la calculatrice.

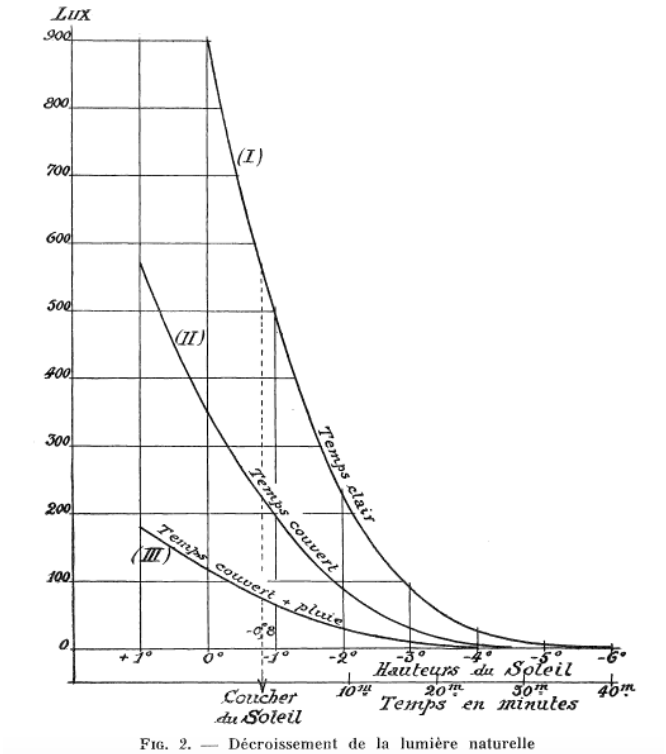


Fig. 2. — Décroissement de la lumière naturelle pendant le crépuscule civil.

Unité 6 : Utiliser les librairies TI Hub & TI Rover

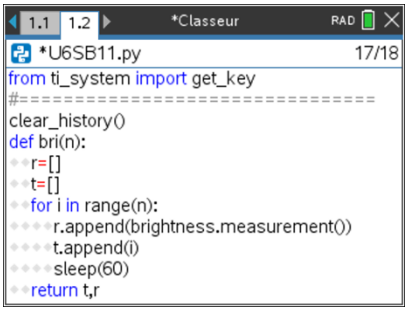
Compétence 1 : Les capteurs intégrés au hub

Dans cette première leçon de l'unité 6, vous allez découvrir comment utiliser la librairie **TI_Hub** afin de commander les dispositifs intégrés au hub TI-Innovator™.

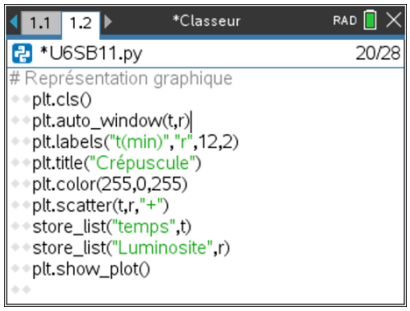
Objectifs :

- Découvrir le module **TI-Hub**.
- Écrire un script intégrant la librairie TI Hub pour les dispositifs intégrés.

a) Acquisition des données



b) Représentation graphique



La fonction **bri(n)** réalise l'acquisition de données toutes les minutes pendant **n** minutes et renvoie les liste **t** et **r**.
La fonction **graphe(t,r)** réalise la représentation graphique des données **t[ ]** et **r[ ],** puis les exporte vers les listes **temps** et **luminosite** de la calculatrice.

Unité 6 : utiliser les bibliothèques TI Hub & TI Rover

Compétence 2 : les dispositifs d'entrée-sortie

Dans cette seconde leçon de l'unité 6, vous allez découvrir comment connecter et utiliser un dispositif d'entrée-sortie du TI-Innovator™ à l'aide de la bibliothèque **TI Hub**.

Objectifs :

- Découvrir le module **TI Hub**.
- Écrire et utiliser un script permettant d'utiliser un composant d'entrée/sortie « grove ».

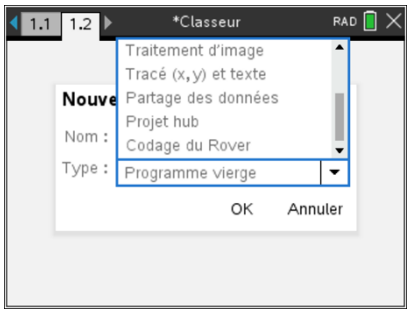
Vous allez dans cette leçon, utiliser un composant essentiel dans toute chaîne de mesures utilisant des capteurs : il s'agit d'un potentiomètre.

Un **potentiomètre** est un type de résistance variable à trois bornes, dont une est reliée à un curseur se déplaçant sur une piste résistante terminée par les deux autres à laquelle est soumise la résistance.
Les potentiomètres sont couramment employés dans les circuits électroniques. Ils servent par exemple à contrôler le volume d'une radio. Les potentiomètres peuvent aussi être utilisés comme des transducteurs, puisqu'ils convertissent une position en une tension. Ce type de dispositif peut être rencontré dans des joysticks.

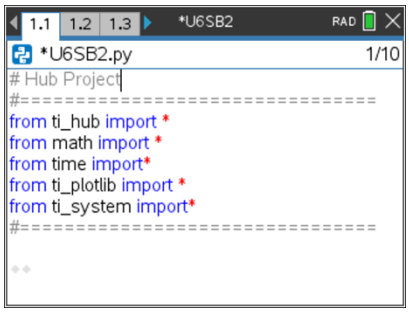
Vous allez écrire un script permettant de mesurer la tension électrique entre deux bornes du potentiomètre, puis de l'afficher à l'écran.

Remarque : L'esprit de cette leçon ne porte pas sur l'étude du composant en lui-même, mais sur son intégration au sein d'un script Python afin d'obtenir les informations que celui-ci doit fournir. Ainsi le script que vous allez réaliser sera aisément transposable à tout autre type de transducteur.

Mise en œuvre :
Commencer un nouveau script et le nommer U6SB2.



Les bibliothèques **TI System** et **Time** sont importées. Vous allez à présent importer la bibliothèque du Hub correspondant au potentiomètre et éventuellement la bibliothèque **TI Plot**.



Unité 6 : utiliser les bibliothèques TI Hub & TI Rover

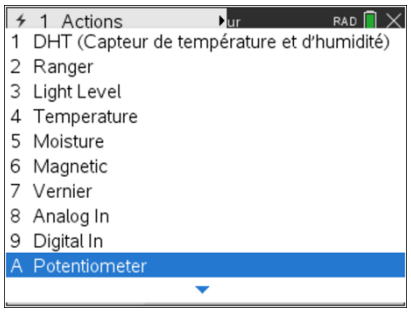
Compétence 2 : les dispositifs d'entrée-sortie

Dans cette seconde leçon de l'unité 6, vous allez découvrir comment connecter et utiliser un dispositif d'entrée-sortie du TI-Innovator™ à l'aide de la bibliothèque **TI Hub**.

Objectifs :

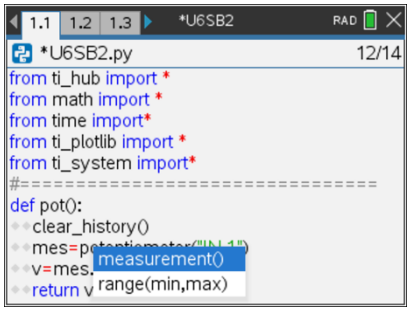
- Découvrir le module **TI Hub**.
- Écrire et utiliser un script permettant d'utiliser un composant d'entrée/sortie « grove ».

- Créer une fonction **pot( )** ne comportant aucun argument.
- Effacer l'écran à l'aide de l'instruction **clear_history()** située dans la bibliothèque **TI System**.
- Dans la bibliothèque TI Hub, choisir le menu **3 Ajouter une unité d'entrée**, puis **A Potentiometer**.
- Compléter l'instruction en affectant l'instruction **potentiometer( )** à la variable **mes**.
- Terminer l'instruction en affectant le potentiomètre au port **In1**.
- L'utilisation de la touche [var] permet d'avoir la complétion automatique des instructions.



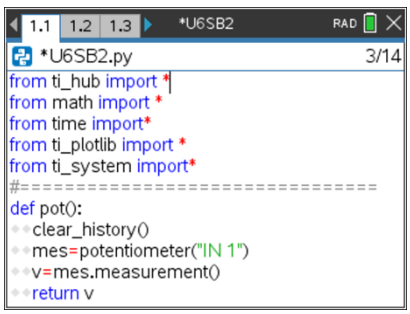
Conseil à l'enseignant : tous les capteurs (**dispositifs d'entrée**) comportent un minimum de deux variables **1 : var=capteur(« port »)** et **2 : var.measurement( )**

- Créer une variable **v** permettant de collecter la mesure du capteur connecté (variable **mes**). Sur la calculatrice, l'auto complétion sera proposée dès que **v=mes**, sera écrit. L'instruction measurement() n'étant pas dans le menu **TI Hub**.



- Vérifier le fonctionnement de votre capteur après avoir préalablement placé le potentiomètre sur une position centrale.
- Connecter le TI-Innovator™ à la calculatrice, puis le potentiomètre au port **IN1**
- Appeler la fonction **pot( )**.

Vous devriez obtenir une information du même ordre de grandeur que celle de l'écran ci-contre, mais constater qu'il ne s'agit pas d'une tension puisque votre potentiomètre est alimenté par une tension de 3.3V. La valeur à trouver appartiendra à l'intervalle [0 ; 3.3].



Unité 6 : utiliser les bibliothèques TI Hub & TI Rover

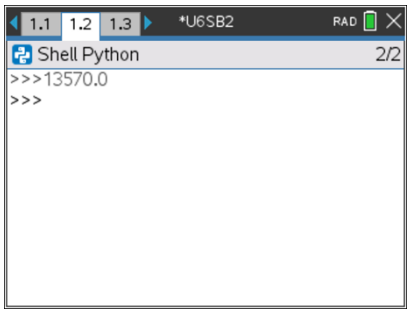
Compétence 2 : les dispositifs d'entrée-sortie

Dans cette seconde leçon de l'unité 6, vous allez découvrir comment connecter et utiliser un dispositif d'entrée-sortie du TI-Innovator™ à l'aide de la bibliothèque **TI Hub**.

Objectifs :

- Découvrir le module **TI Hub**.
- Écrire et utiliser un script permettant d'utiliser un composant d'entrée/sortie « grove ».

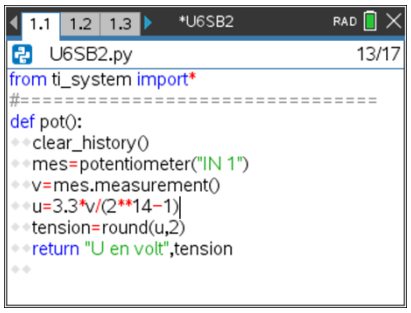
- Exécuter le script puis appeler la fonction **pot( )**.



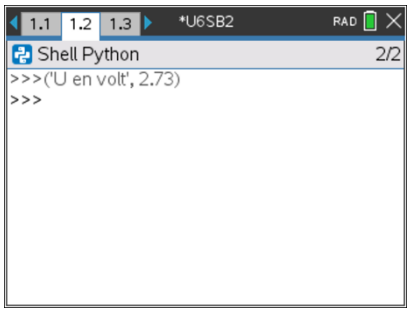
- Modifier votre script afin de prendre en compte la résolution du convertisseur analogique numérique (14 bits). Ainsi la mesure de la tension sera :

$$u = U_{alim} \times \frac{v}{2^{14} - 1}$$

- La tension sera arrondie à 10^{-2} près.



- Exécuter le script et effectuer plusieurs mesures.



Quelques idées pour prolonger la leçon :

- Utiliser un potentiomètre pour réaliser un capteur angulaire (une mesure de tension correspond à la valeur d'un angle lue sur un rapporteur), puis réaliser la représentation graphique de la fonction modélisée $\alpha = f(u)$.
- Entre 0 et 3.3v, associer une plage de tension à une couleur en utilisant la DEL RVB du TI-Innovator™.
- Associer la mesure de la tension aux coordonnées d'un point repéré (principe d'un joystick).
- …etc.

Unité 6 : utiliser les bibliothèques TI Hub & TI Rover

Compétence 3 : les dispositifs d'entrée-sortie

Dans cette seconde leçon de l'unité 6, vous allez découvrir comment connecter le ti-rover à l'aide de la bibliothèque **TI Rover**.

Objectifs :

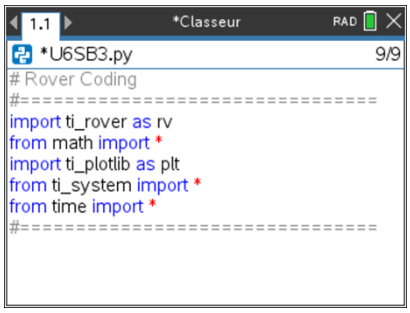
- Découvrir le module **TI Rover**.
- Écrire et utiliser un script permettant d'utiliser TI-Innovator™ Rover et ses actionneurs associés.
- Utiliser une boucle ouverte et une instruction conditionnelle.

Vous allez, dans cette leçon, réaliser un script donnant au TI-Innovator™ Rover la possibilité d'effectuer un parcours marqué par l'illumination de la diode RVB, tant que la distance (mesurée par le capteur RANGER) respecte une limite inscrite dans une instruction conditionnelle.

```
Avancer sur une distance de 2m
    Tant que le mouvement n'est pas stoppé par l'utilisateur
        a ⟵ distance à un objet mesurée
                si a < 0.2
                        alors afficher une couleur rouge et s'arrêter
                        sinon afficher une couleur verte et continuer
    S'arrêter, afficher une couleur bleue
    Attendre 1s
    Eteindre la diode
Allumer la diode en bleu pour marquer la fin
```

- Commencer un nouveau script et le nommer U6SB3.
- Choisir la bibliothèque « **Codage du Rover** ».
- Valider.

- Vous êtes maintenant prêts à écrire votre script.



Unité 6 : utiliser les bibliothèques TI Hub & TI Rover

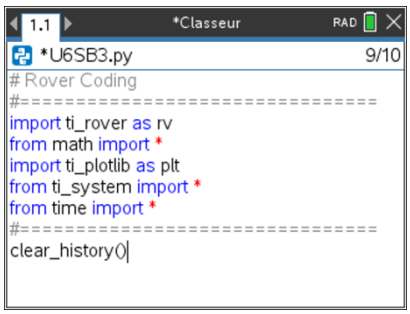
Compétence 3 : les dispositifs d'entrée-sortie

Dans cette seconde leçon de l'unité 6, vous allez découvrir comment connecter le ti-rover à l'aide de la bibliothèque **TI Rover**.

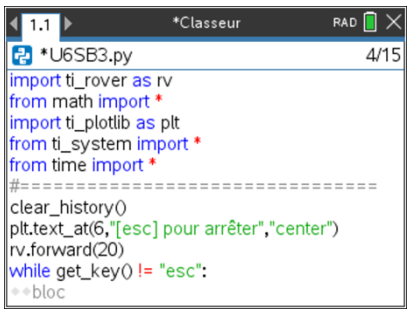
Objectifs :

- Découvrir le module **TI Rover**.
- Écrire et utiliser un script permettant d'utiliser TI-Innovator™ Rover et ses actionneurs associés.
- Utiliser une boucle ouverte et une instruction conditionnelle.

- Effacer votre écran à l'aide de l'instruction **clear_history()**, située dans le menu **TI System**.
- Demander au TI-Innovator™ Rover de se déplacer en avant. L'unité de mesure de la distance est laissée à votre choix sachant que par défaut, celle-ci est fixée à 0,1 m. Ainsi **rv.forward(20)** assignera au robot un déplacement en avant sur une distance de 2 m. L'instruction **rv.forward( )** est située dans le menu **TI Rover** et enfin **2 Lecteur**.
- Inscrire ensuite, le début d'une boucle ouverte que l'on trouve dans le menu de la bibliothèque **TI System**.



Conseil à l'enseignant : Un grand nombre d'instructions disponibles dans le menu de la bibliothèque **TI System**, le sont également dans celui de la bibliothèque **TI Rover** sous le menu **Commandes**.



Unité 6 : utiliser les bibliothèques TI Hub & TI Rover

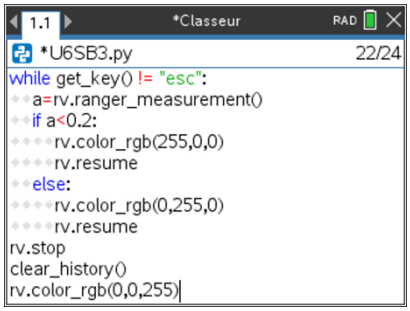
Compétence 3 : les dispositifs d'entrée-sortie

Dans cette seconde leçon de l'unité 6, vous allez découvrir comment connecter le ti-rover à l'aide de la bibliothèque **TI Rover**.

Objectifs :

- Découvrir le module **TI Rover**.
- Écrire et utiliser un script permettant d'utiliser TI-Innovator™ Rover et ses actionneurs associés.
- Utiliser une boucle ouverte et une instruction conditionnelle.

- Créer une variable **a** à laquelle est affectée la distance mesurée par le RANGER. Pour cela, commencer à écrire la lettre **a**, puis laisser le curseur à la fin de cette lettre. Inscrire ensuite l'instruction **rv.ranger_measurement( )** située dans le menu **9 TI Rover** puis **3 Entrées** puis **E/S** et enfin **1 rv.ranger_measurement( ).** L'unité de mesure est le mètre.
- Créer à présent l'instruction conditionnelle. Si la distance mesurée est inférieure à 20 cm, le robot s'arrête et la diode RVB s'allume en rouge. L'instruction **rv.color( )** est disponible dans la bibliothèque **TI Rover** au menu **4 sortie**.
- **rv.stop( )** est une instruction de conduite et donc placée sous le menu correspondant. Sinon la diode RVB est de couleur verte, et le robot poursuit son parcours jusqu'à attendre la distance fixée. L'instruction **rv.resume( )** termine le traitement des actions en court dans la file d'attente.



- A la fin de la boucle :
  - Le robot s'arrête : **rv.stop( ).**
  - L'écran est effacé.
  - La diode affiche une couleur bleue.

Un délai d'attente de 1s précède l'extinction de la diode.

Unité 6 : utiliser les bibliothèques TI Hub & TI Rover

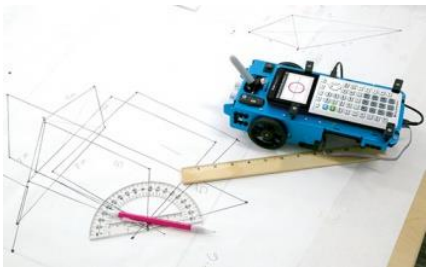
Application : Représenter un parcours

Dans cette application de l'unité 6, vous allez connecter le TI-Innovator™ Rover à l'aide de la bibliothèque **TI Hub** et construire un script permettant d'enregistrer des coordonnées de points lors d'un parcours, puis de les représenter graphiquement.

Objectifs :

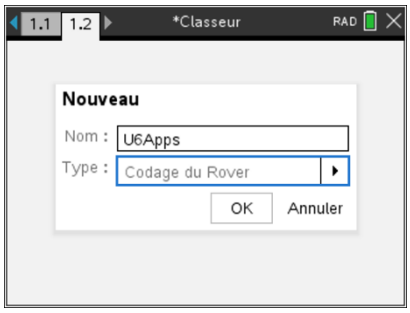
- Découvrir le module **TI Rover**.
- Écrire et utiliser un script permettant d'utiliser le TI-Innovator™ Rover et ses actionneurs associés.
- Utiliser une boucle fermée.
- Représenter graphiquement des données.

Vous allez, dans cette leçon, réaliser un script donnant au TI-Innovator™ Rover la possibilité d'effectuer un parcours correspondant au dessin d'un polygone.
Les coordonnées des sommets seront sauvegardées dans des listes puis représentées graphiquement à l'aide des instructions de la bibliothèque **TI Plot**.

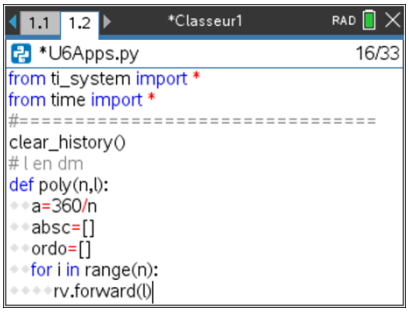


Mise en œuvre :

- Commencer un nouveau script et le nommer U6APPS.
- Choisir un script de type « Codage du Rover ».
- Valider en appuyant sur la touche enter.
- Importer également la bibliothèque **TI Plotlib**



- Bien que cela ne soit pas impérativement nécessaire, les instructions permettant d'effacer l'écran et de ne pas afficher le curseur sont toujours d'un rendu plus agréable. Insérer l'instruction **clear_history()** se trouvant dans la bibliothèque **TI System**.
- Créer une fonction **poly(),** prenant en argument n le nombre de côtés du polygone et l la longueur en unité par défaut pour le TI-Innovator™ Rover, c'est-à-dire le dm.
- L'angle au sommet de chaque polygone sera donc égal $a = \frac{360}{n}$.
- Créer deux listes vides **absc** et **ordo**, destinées à recevoir les coordonnées de chacun de ces sommets.



Unité 6 : utiliser les bibliothèques TI Hub & TI Rover

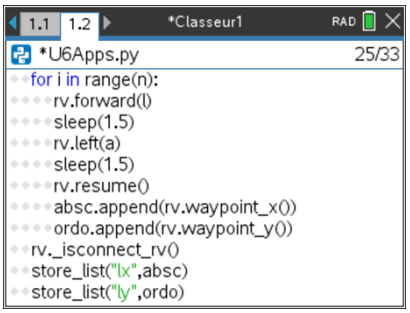
Application : Représenter un parcours

Dans cette application de l'unité 6, vous allez connecter le TI-Innovator™ Rover à l'aide de la bibliothèque **TI Hub** et construire un script permettant d'enregistrer des coordonnées de points lors d'un parcours, puis de les représenter graphiquement.

Objectifs :

- Découvrir le module **TI Rover**.
- Écrire et utiliser un script permettant d'utiliser le TI-Innovator™ Rover et ses actionneurs associés.
- Utiliser une boucle fermée.
- Représenter graphiquement des données.

- Créer une boucle ouverte de taille n.

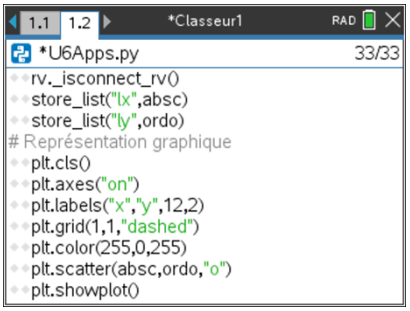


Les instructions suivantes se trouvent dans la bibliothèque **TI Rover**.

- Faire avancer le rover de $\boldsymbol{l}$ décimètres pour $\boldsymbol{n}$ fois.
- Tourner sur la gauche d'un angle $\boldsymbol{a}$.
- Mettre entre chaque étape un délai de 1,5s.
- Stocker dans les listes x et y les coordonnées des points.
- Déconnecter le Rover à la fin du parcours.
- Exporter les coordonnées **absc** et **ordo** respectivement dans les listes **lx** et **ly**, pour éventuellement faire une représentation graphique des données dans une application de la TI-Nspire™ (Graphiques, Tableur&Listes…).

Conseil à l'enseignant : les instructions **rv.waypoint_x( )** et **rv.waypoint_y( )** se trouvent dans le menu **5 Path** de la bibliothèque **TI Rover**.

- Passer ensuite à la représentation graphique ; celle-ci est ici incluse dans la fonction, mais il est toujours possible de créer une fonction graphe séparément comme nous avons pu le faire dans les leçons précédentes (Unité 5).



Unité 6 : utiliser les bibliothèques TI Hub & TI Rover

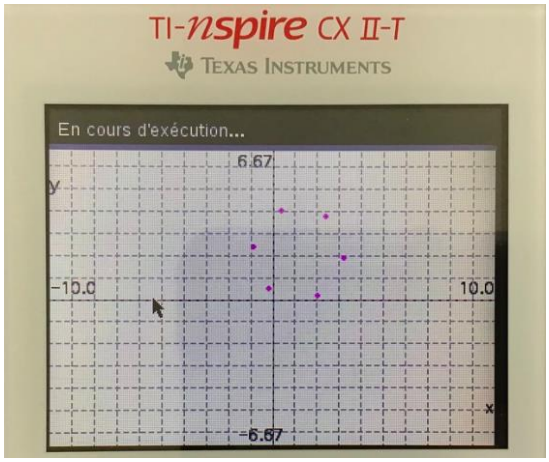
Application : Représenter un parcours

Dans cette application de l'unité 6, vous allez connecter le TI-Innovator™ Rover à l'aide de la bibliothèque **TI Hub** et construire un script permettant d'enregistrer des coordonnées de points lors d'un parcours, puis de les représenter graphiquement.

Objectifs :

- Découvrir le module **TI Rover**.
- Écrire et utiliser un script permettant d'utiliser le TI-Innovator™ Rover et ses actionneurs associés.
- Utiliser une boucle fermée.
- Représenter graphiquement des données.

Exécuter votre script et appeler la fonction **poly( )** en lui fournissant par exemple les arguments poly(4, 1), afin de dessiner dans un premier temps un carré de 1 dm de côté.



Poursuivre par un essai avec un hexagone de 2 dm ce qui fournit la représentation suivante.

Conseil à l'enseignant : Pour ce type d'exercice, éviter d'utiliser les instructions **rv.pathlist_x( )** et **rv.pathlist_y( )**.
En fait lors du tracé d'un segment, la calculatrice enregistre les coordonnées des points d'un segment du polygone, puis réenregistre les coordonnées du dernier point comme premier point du segment suivant. D'autant plus que nous avons placé, entre chaque tracé, une temporisation de 1s.
Ainsi les instruction rv.path… sont dans notre cas inappropriées.
En utilisant les instructions **rv.pathlist_x( )** et **rv.pathlist_y( ),** on obtiendrait deux fois les coordonnées des extrémités.

Remarque : Ajuster le format de votre grille, en fonction des polygones que vous souhaitez tracer. Celle-ci a volontairement été réglée ici aux paramètres par défaut, afin d'observer la précision du tracé du TI-Innovator™ Rover.

Prendre garde également à la nature de la surface sur laquelle se déplace le robot. Celle-ci ne doit pas opposer trop de résistance au déplacement ou, au contraire, favoriser les glissements.

Unité 7 : Utiliser la bibliothèque cmath

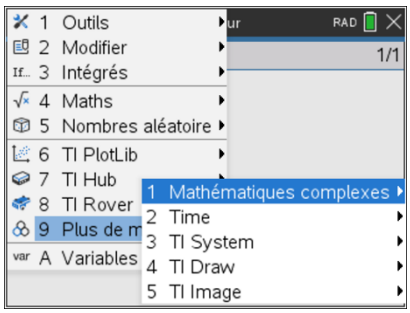
Compétence 1 : Les fonctions de la bibliothèque cmath

Dans cette première leçon de l'unité 7, vous allez découvrir comment utiliser la bibliothèque **cmath** pour effectuer des calculs simples sur les nombres complexes.

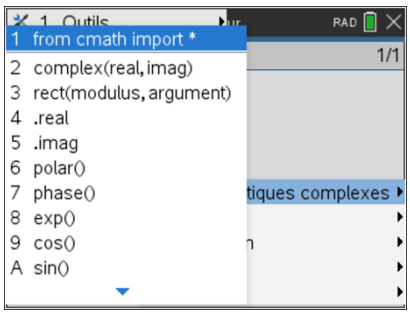
Objectifs :

- Découvrir la bibliothèque **cmath**.
- Utiliser les fonctions de la bibliothèque **cmath**.

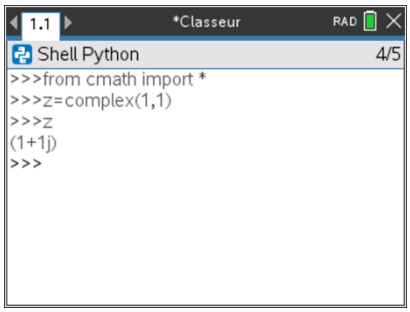
1. Utiliser le module **cmath.**
   - Insérer une nouvelle application et choisir le menu **A Ajouter Python**.
   - Dans cette leçon, nous allons essentiellement travailler en utilisant la console (shell), afin de découvrir les instructions de la bibliothèque **cmath**.
   - Dans la fenêtre qui s'ouvre, choisir l'option **3 Shell**.
   - La touche menu donne accès à **9 Plus de modules**, puis **1 Mathématiques complexes**.



   - Importer la bibliothèque **cmath**.
   - Choisir l'option **2 complex(real,imag)** et affecter ce nombre à une variable **z**.



   - Demander l'affichage de **z**.
   - La calculatrice utilise **j** pour désigner le nombre imaginaire pur.



On remarquera que le nombre complexe est édité entre parenthèse sous la forme

$$z = a + bj$$

   - De la même façon, définir un nombre complexe sans passer par l'instruction **complex(real, imag)**, mais directement en utilisant la même syntaxe, soit par exemple $z2 = (2 - 3j)$

Conseil à l'enseignant : l'oubli de parenthèses lors de l'écriture d'un nombre complexe dans la console entraîne l'édition d'un message d'erreur.

Unité 7 : Utiliser la bibliothèque cmath

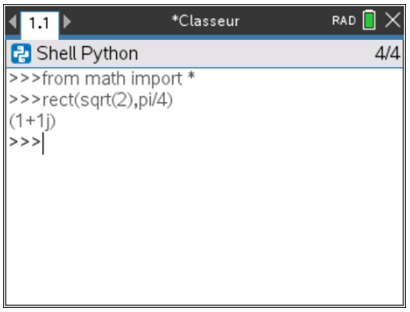
Compétence 1 : Les fonctions de la bibliothèque cmath

Dans cette première leçon de l'unité 7, vous allez découvrir comment utiliser la bibliothèque **cmath** pour effectuer des calculs simples sur les nombres complexes.

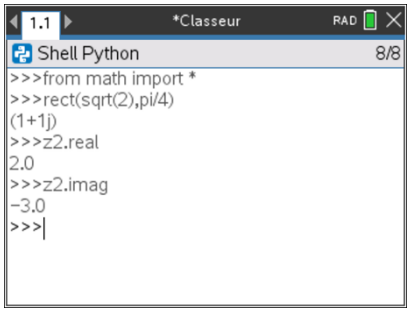
Objectifs :

- Découvrir la bibliothèque **cmath**.
- Utiliser les fonctions de la bibliothèque **cmath**.

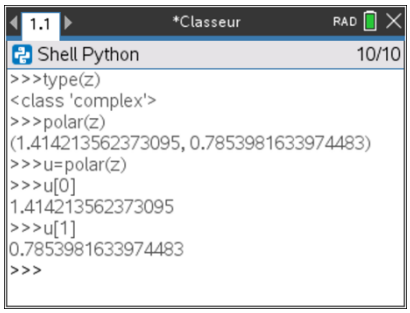
- Importer également dans la console, la bibliothèque de calculs mathématiques.
- Dans la bibliothèque **cmath**, choisir l'instruction **3 rect(module, argument)**.
- Compléter l'instruction rect(sqrt(2),pi/4), soit la demande en coordonnées rectangulaires d'un nombre complexe de module $\sqrt{2}$ et d'argument $\frac{\pi}{4}$.



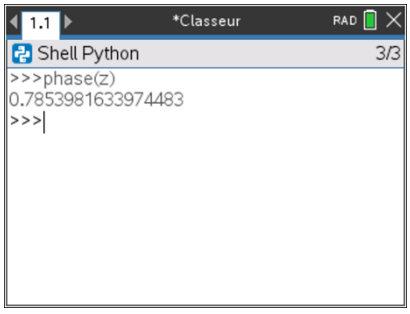
- L'affichage des parties réelles et imaginaires d'un nombre complexe s'effectue par l'intermédiaire des méthodes **.real** et **.imag** précédées du nom de la variable complexe.



- L'instruction **6 polar()** ne doit comporter comme seul argument que le nom de la variable complexe afin de renvoyer un tuple dont le premier élément sera le module du nombre complexe et le second un argument.



- L'instruction **7 phase()** donne l'argument du nombre complexe exprimé en radians.



Unité 7 : Utiliser la bibliothèque cmath

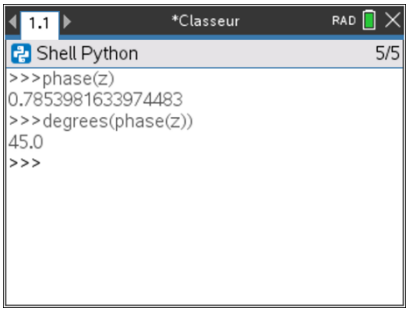
Compétence 1 : Les fonctions de la bibliothèque cmath

Dans cette première leçon de l'unité 7, vous allez découvrir comment utiliser la bibliothèque **cmath** pour effectuer des calculs simples sur les nombres complexes.

Objectifs :

- Découvrir la bibliothèque **cmath**.
- Utiliser les fonctions de la bibliothèque **cmath**.

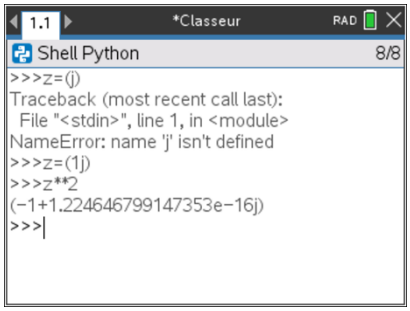
- Pour exprimer un argument d'un nombre complexe en degrés, utiliser l'instruction **degrees** issue de la bibliothèque de fonctions mathématiques.

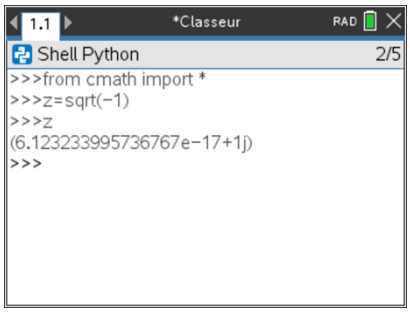


Conseil à l'enseignant : Le module d'un nombre complexe peut également être obtenu en utilisant l'instruction abs(z).

Carré d'un imaginaire pur.

- Créer le nombre complexe $z = j$ (attention à l'instruction à fournir à la calculatrice).
- Calculer z^2
- On obtient bien le résultat attendu, avec la précaution d'usage de bien conserver à l'esprit la façon dont les nombres décimaux sont exprimés en langage python.
- On pourra écrire dans un script une fonction permettant d'affecter 0 à la partie réelle ou imaginaire d'un nombre complexe, lorsque sa valeur n'excède pas 10^{-n}, la valeur de n étant à préciser.





Unité 7 : Utiliser la bibliothèque cmath

Compétence 1 : Les fonctions de la bibliothèque cmath

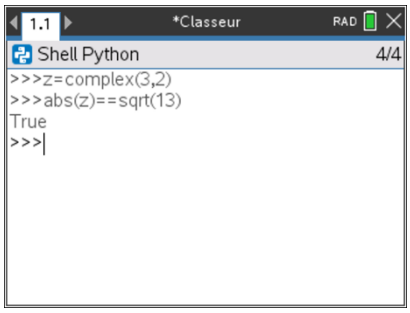
Dans cette première leçon de l'unité 7, vous allez découvrir comment utiliser la bibliothèque **cmath** pour effectuer des calculs simples sur les nombres complexes.

Objectifs :

- Découvrir la bibliothèque **cmath**.
- Utiliser les fonctions de la bibliothèque **cmath**.

2. Quelques calculs élémentaires.

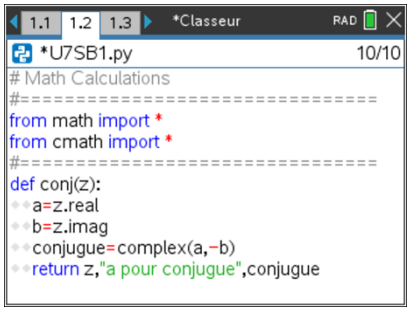
a) Définir le nombre complexe de partie réelle 3 et de partie imaginaire 2. Vérifier que le module de ce nombre est $\sqrt{13}$.



b) Nombre conjugué d'un nombre complexe.

Le conjugué d'un nombre complexe n'est pas implémenté dans la calculatrice TI-Nspire™. On se propose donc d'écrire pour terminer cette leçon, une fonction qui renvoie le conjugué d'un nombre complexe donné.

- Insérer une nouvelle application Python permettant d'écrire un script et le nommer U7SB1.
- Importer les bibliothèques de calculs mathématiques et le module **cmath**.
- Extraire les parties réelle et imaginaire du complexe passé en argument de la fonction.
- Afficher les deux nombres z et $\bar{z}$.



Exemple : Soit $z = 2 + 3j$, déterminer à l'aide de la fonction **conj(z)** le nombre complexe conjugué.

Unité 7 : Utiliser la bibliothèque cmath

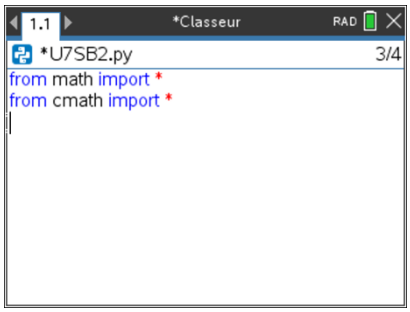
Compétence 2 : Calculs et représentations

Dans cette seconde leçon de l'unité 7, vous allez utiliser la bibliothèque **cmath** pour effectuer des calculs simples sur les nombres complexes.

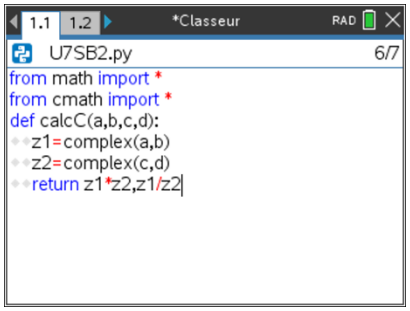
Objectifs :

- Utiliser la bibliothèque **cmath**.
- Réaliser des calculs sur les nombres complexes.
- Représenter graphiquement des nombres complexes.

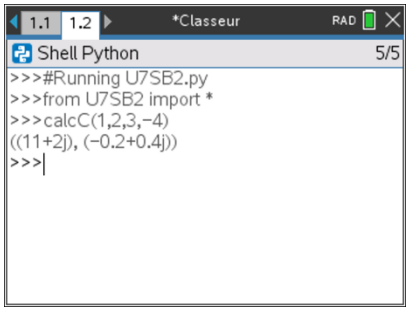
1. Quelques calculs simples**.**
   - Commencer un nouveau script et le nommer U7SB2.
   - Insérer une nouvelle application et choisir le menu **A Ajouter Python**.
   - La touche menu donne accès à **9 Plus de modules**, puis **1 Mathématiques complexes**.
   - Importer également la bibliothèque de fonctions mathématiques.



   - Créer une fonction **calcC(a,b,c,d)** ayant pour arguments respectivement, les parties réelles et imaginaires des nombres complexes : $z1 = a + bj$ et $z2 = c + jd$
   - Demander à cette fonction de retourner le produit et le quotient de ces deux nombres, soient $z1 \times z2$ et $\frac{z1}{z2}$.



   - Tester la fonction avec deux nombres complexes de votre choix.



2. Les diverses expressions d'un nombre complexe.

Vous allez maintenant créer une fonction permettant de travailler plus simplement sur les nombres complexes en utilisant les formes trigonométriques et exponentielles.

Unité 7 : Utiliser la bibliothèque cmath

Compétence 2 : Calculs et représentations

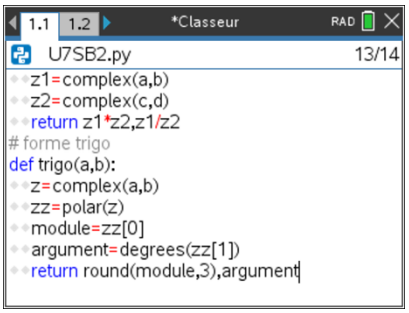
Dans cette seconde leçon de l'unité 7, vous allez utiliser la bibliothèque **cmath** pour effectuer des calculs simples sur les nombres complexes.

Objectifs :

- Utiliser la bibliothèque **cmath**.
- Réaliser des calculs sur les nombres complexes.
- Représenter graphiquement des nombres complexes.

a) Forme trigonométrique et exponentielle.

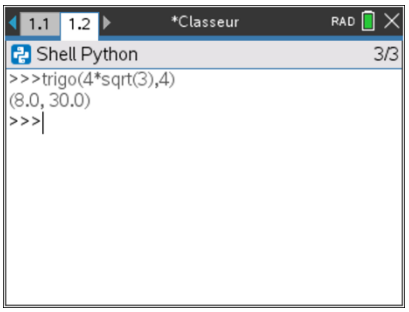
Ecrire une fonction permettant de donner le module et un argument d'un nombre complexe (radian et degrés) afin de pouvoir l'écrire sous la forme $z = \rho \times (cos\theta + jsin\theta)$ puis $z = \rho \times e^{j\theta}$.



Conseil à l'enseignant : le module et un argument d'un nombre complexe peuvent également être déterminés en utilisant les instruction **abs()** et **phase()** de la bibliothèque **cmath**.

Etude d'un exemple.
Soit le nombre complexe $z = 4\sqrt{3} + 4j$

- Déterminer le module et un argument de ce nombre complexe (mod 2π).

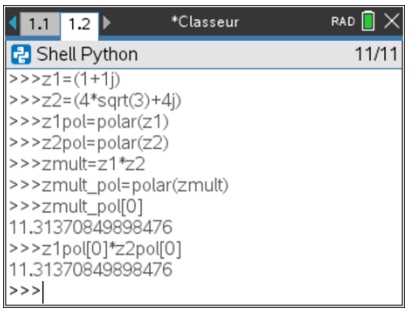


Donner l'expression de sa forme trigonométrique puis exponentielle.
La fonction permet rapidement de vérifier que le nombre complexe a pour module $\rho = 8$ et pour argument $\theta = 30°$ soit $\frac{\pi}{6}$ mod 2π.
La forme exponentielle sera $z = 8 \times e^{j\frac{\pi}{6}}$.

b) Intérêt des formes trigonométriques et exponentielles.

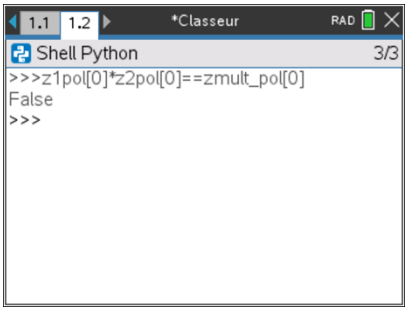
Utiliser les deux fonctions précédentes (ou bien en créer une autre utilisant les deux précédentes), afin de vérifier que pour deux nombres complexes :

- Le module du produit, est le produit des modules et un argument du produit, est la somme des arguments.
- Le module du quotient, est le quotient des modules et un argument du quotient, est la différence des arguments.



On peut également travailler directement dans la console.
Etude d'un exemple : $z1 = 1 + 1j$ et $z2 = 4\sqrt{3} + 4j$

Puis faire ensuite appel aux opérateurs booléens, **mais attention** !



Unité 7 : Utiliser la bibliothèque cmath

Compétence 2 : Calculs et représentations

Dans cette seconde leçon de l'unité 7, vous allez utiliser la bibliothèque **cmath** pour effectuer des calculs simples sur les nombres complexes.

Objectifs :

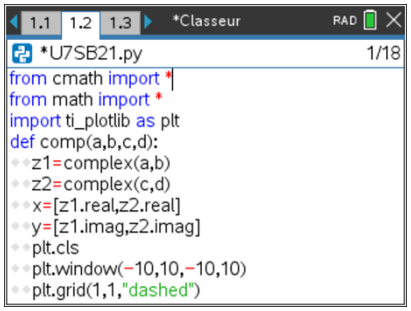
- Utiliser la bibliothèque **cmath**.
- Réaliser des calculs sur les nombres complexes.
- Représenter graphiquement des nombres complexes.

3. Représenter graphiquement un nombre complexe.

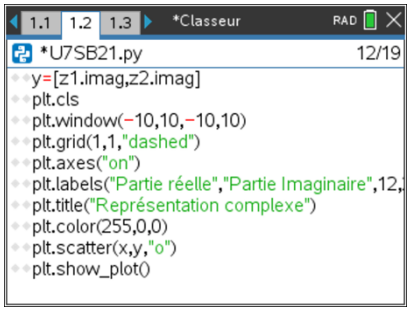
Vous allez représenter dans le plan, les nombres complexes précédents :
$z1 = 1 + 1j$ et $z2 = 4\sqrt{3} + 4j$
Pour cela vous devez :

- Extraire les parties réelles et imaginaires des nombres complexes.
- Les stocker dans deux listes $x[\ \ ]$ et $y[\ \ ]$.
- Représenter graphiquement ces listes sous forme d'un nuage de points.

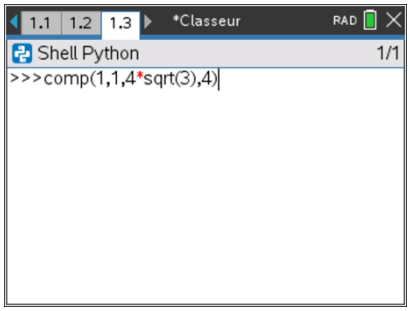
Insérer un nouveau script Python et le nommer U7SB21.



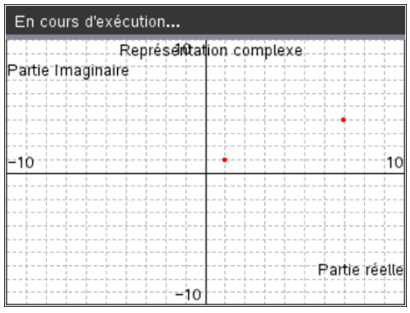
Créer une fonction permettant d'effectuer la représentation graphique des deux nombres complexes. Cette fonction peut paraitre artificielle afin de représenter graphiquement deux complexes d'affixe z. C'est un premier pas vers la leçon suivante (Compétence 3), au cours de laquelle vous effectuerez la résolution d'une équation complexe.



- Exécuter votre script.
- Demander la représentation graphique des nombres proposés.



- Si vous le souhaitez, vous pouvez modifier la représentation graphique, afin de mettre en évidence le module et un argument (importation éventuelle de la bibliothèque **TI Draw**).



Unité 7 : Utiliser la bibliothèque cmath

Compétence 3 : Représenter des nombres complexes

Dans cette troisième leçon de l'unité 7, vous utiliser la bibliothèque **cmath** associée à la bibliothèque **TI PlotLib** pour effectuer des représentations de nombres complexes.

Objectifs :

- Découvrir le module **cmath**.
- Utiliser les fonctions de la bibliothèque **cmath**.
- Représenter graphiquement des nombres complexes.

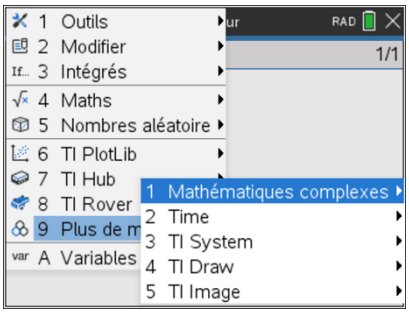
Écriture complexe d'une transformation géométrique.

Une transformation F du plan transforme chaque point M en son image M'. Aux points M et M', on associe respectivement leurs affixe z et z'.
L'écriture complexe de la transformation F est : $z' = f(z)$ où f est la fonction de $\mathbb{C} \to \mathbb{C}$ qui à z associe z'.
L'écriture complexe d'une rotation de centre Ω d'affixe ω et d'angle θ est :

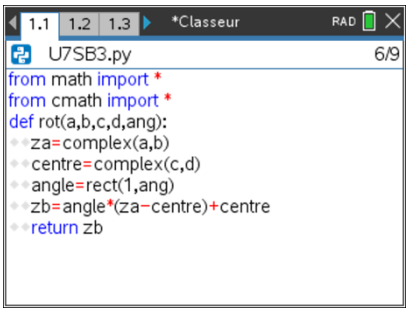
$$z' = e^{i\theta}(z - \omega) + \omega$$

Déterminer l'affixe z_B de l'image du point A d'affixe $z_A = 1 + 2j$ par la rotation d'angle $\frac{2\pi}{3}$ et de centre de point d'affixe $\omega = -1 + j$

- Insérer une nouvelle application et choisir le menu **A Ajouter Python**.
- Créer un nouveau script et le nommer U7SB3
- La touche menu donne accès à **9 Plus de modules**, puis **1 Mathématiques complexes**.
- Insérer les bibliothèques **math** et **cmath**



- Vous allez créer une fonction comportant 3 arguments et permettant de déterminer l'affixe $z_B = c + dj$ de l'image d'un point A d'affixe $z_A = a + bj$ par une rotation d'angle θ et de centre Ω d'affixe ω.



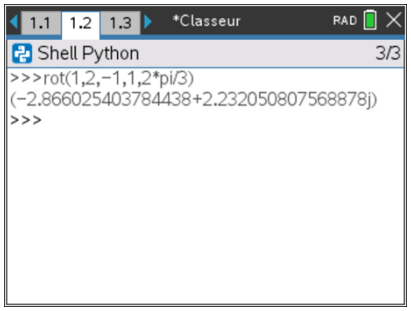
Unité 7 : Utiliser la bibliothèque cmath — Compétence 3 : Représenter des nombres complexes

Dans cette troisième leçon de l'unité 7, vous utiliser la bibliothèque **cmath** associée à la bibliothèque **TI PlotLib** pour effectuer des représentations de nombres complexes.

Objectifs :

- Découvrir le module **cmath**.
- Utiliser les fonctions de la bibliothèque **cmath**.
- Représenter graphiquement des nombres complexes.

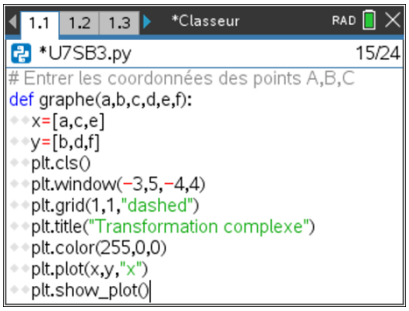
- Exécuter le script.
- Vérifier que l'affixe de z_B est : $z_B = \frac{-4-\sqrt{3}}{2} + \frac{1+2\sqrt{3}}{2} \times j$



Transformation complexe et représentation graphique.

Vous allez maintenant réinvestir l'utilisation de la fonction précédente afin de montrer qu'un triangle est équilatéral.

Soient $A(a,b), B(c,d), C(e,f)$ les points d'affixes respectives : $za = \sqrt{3} + 2 - 3j$; $zb = -2$ et $zc = 2\sqrt{3} + 2j\sqrt{3}$



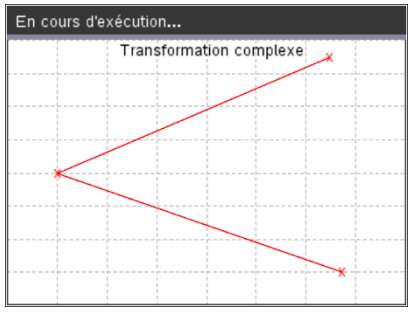
- Modifier le script afin que celui-ci effectue la représentation graphique. Pour cela, créer deux listes x et y contenant respectivement les parties réelles et imaginaires des complexes $za, zb, et\ zc$.
- Représenter graphiquement le nuage de trois points.
- Utiliser la fonction **rot( )** afin de montrer par exemple que le point A est l'image du point C par la rotation *r* de centre B et d'angle $-\frac{\pi}{3}$.
- L'écriture complexe de *r* est donc $z' = e^{-j\frac{\pi}{3}}(z-b) + b$

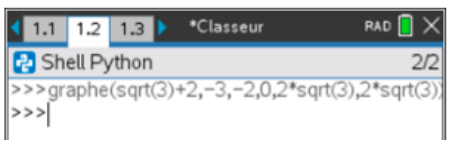
D'où $c' = \left(\frac{1}{2} - \frac{\sqrt{3}}{2}j\right)\left(2\sqrt{3} + 2j\sqrt{3} + 2\right) - 2$ soit $c' = \sqrt{3} + 1 + \sqrt{3}j - 3j - j\sqrt{3} + 3 - 2$

Soit $c' = \sqrt{3} + 2 - 3j$

A est l'image de C par *r*, ce qui donne BC = BA et $\left(\overrightarrow{BC}; \overrightarrow{BA}\right) = -\frac{\pi}{3}\ [2\pi]$.

- Exécuter le script
- Utiliser la fonction **rot( )** en calculant l'affixe de C, image de A par la rotation de centre $\omega = zb$ et d'angle $-\frac{\pi}{3}$.





Unité 7 : Utiliser la bibliothèque cmath

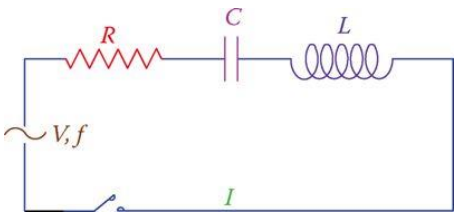
Application : Nombres complexes et sciences

Dans cette application de l'unité 7, vous allez utiliser la bibliothèque **cmath** pour effectuer des calculs et représenter des nombres complexes utilisés en sciences physiques pour l'étude d'un circuit électrique.

Objectifs :

- Découvrir le module **cmath**.
- Utiliser les fonctions de la bibliothèque **cmath**.
- Représenter graphiquement des nombres complexes.
- Analyser un circuit électrique RLC série.

Le circuit RLC série.



Un **circuit RLC** en électrocinétique est un circuit linéaire contenant une résistance électrique, une bobine (inductance) et un condensateur (capacité).

Il existe deux types de circuits **RLC** *série* ou *parallèle*, selon l'interconnexion des trois types de composants. Le comportement d'un circuit RLC est généralement décrit par une équation différentielle du second ordre (là ou des circuits RL ou RC modélisés par des équations différentielles du premier ordre).

A l'aide d'un générateur de signaux, on peut injecter dans le circuit des oscillations et observer dans certains cas une résonance, caractérisée par une augmentation du courant (lorsque le signal d'entrée choisi correspond à la pulsation propre du circuit, calculable à partir de l'équation différentielle qui le régit).

Dans un dipôle linéaire, pas forcément élémentaire, mais constitué d'un assemblage d'éléments linéaires passifs R,L,C si l'on prend l'équation qui lie la tension au courant et que l'on y applique une tension $\bar{U} = U \times e^{j(\omega t - \varphi)}$, on obtiendra un courant $\bar{I} = I \times e^{j(\omega t - \varphi - \psi)}$. On appelle impédance complexe du dipôle la quantité :

$$\bar{Z} = \frac{\bar{U}}{\bar{I}} = Z \times e^{\psi}$$

A quoi sert la notion d'impédance ?

Si on connaît le module Z et un argument φ du dipôle, on peut immédiatement passer de la tension au courant ou réciproquement : le module Z indique le rapport entre l'amplitude de la tension et celle du courant.

Un argument φ donne quant à lui le déphasage entre la tension et le courant.

ω représente la pulsation du signal électrique.

Pour rappel $\omega = 2\pi f$; f étant la fréquence du signal exprimée en Hertz (Hz).

Unité 7 : Utiliser la bibliothèque cmath

Application : Nombres complexes et sciences

Dans cette application de l'unité 7, vous allez utiliser la bibliothèque **cmath** pour effectuer des calculs et représenter des nombres complexes utilisés en sciences physiques pour l'étude d'un circuit électrique.

Objectifs :

- Découvrir le module **cmath**.
- Utiliser les fonctions de la bibliothèque **cmath**.
- Représenter graphiquement des nombres complexes.
- Analyser un circuit électrique RLC série.

Impédance complexe.

Résistor de résistance R : $\bar{Z} = R$

Bobine d'inductance L : $\bar{Z} = jL\omega$

Condensateur de capacité C : $\bar{Z} = \frac{1}{jC\omega}$ ou bien $\bar{Z} = \frac{-j}{C\omega}$

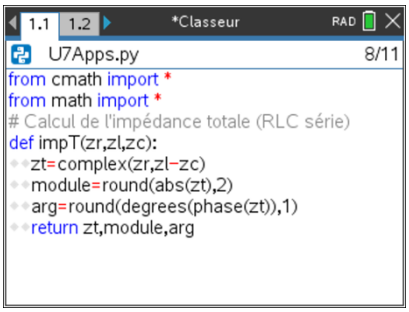
Conseil à l'enseignant : Une impédance se mesure en ohms (Ω). D'un point de vue physique, on s'intéresse au module de l'impédance. Le déphasage introduit par une inductance pure est : $\varphi = \frac{\pi}{2}$ et celui introduit par un condensateur pur est : $\varphi = -\frac{\pi}{2}$.

Etude d'un exemple.

- Créer un script Python afin de déterminer l'impédance complexe d'un circuit RLC série.
- Représenter graphiquement l'impédance de chaque dipôle, puis l'impédance totale.
- En déduire la nature du circuit (dominante inductive ou capacitive).

Rappel : Pour des dipôles disposés en série, leurs impédances s'ajoutent. Lorsque les dipôles sont en parallèle, ce sont leurs admittances Y qui s'ajoutent. $Y = \frac{1}{Z}$. (L'admittance est l'inverse de l'impédance).

- Insérer une nouvelle application et choisir le menu **A Ajouter Python**.
- Créer un nouveau script et le nommer U7Apps.
- Importer les bibliothèques **math** et **cmaths**.
- Créer une fonction comportant 3 arguments, lesquels étant les valeurs des impédances de chaque dipôle dans l'ordre Z_R, Z_L et Z_C.
- La fonction devra retourner l'impédance complexe totale, le module de l'impédance totale, un argument, arrondi au dixième de degré.
-



Conseil à l'enseignant : Si vous le souhaitez, vous pouvez également créer une fonction tenant compte de la fréquence du signal. La fonction aura alors directement pour arguments, les valeurs des dipôles R, L et C, respectivement exprimés en ohms (Ω), henry (H) et farads (F).

Unité 7 : Utiliser la bibliothèque cmath

Application : Nombres complexes et sciences

Dans cette application de l'unité 7, vous allez utiliser la bibliothèque **cmath** pour effectuer des calculs et représenter des nombres complexes utilisés en sciences physiques pour l'étude d'un circuit électrique.

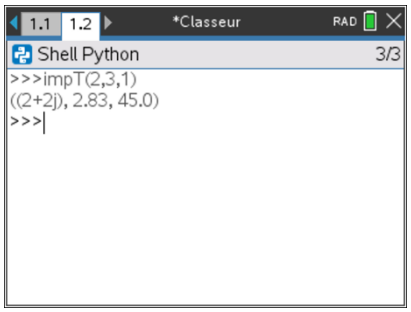
Objectifs :

- Découvrir le module **cmath**.
- Utiliser les fonctions de la bibliothèque **cmath**.
- Représenter graphiquement des nombres complexes.
- Analyser un circuit électrique RLC série.

- Exécuter le script et déterminer l'impédance totale d'un circuit RLC série tel que :

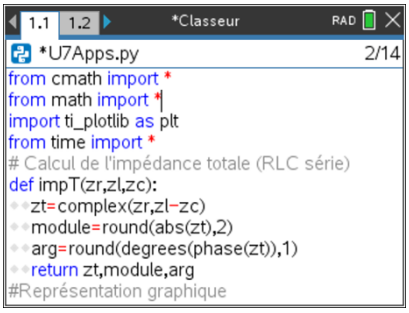
$$zr = 2\Omega\,;\ zl = 3\Omega\,;\ \mathrm{zc} = 1\Omega$$

- La phase est de 45°, le comportement du circuit est à dominante inductive.



Représenter graphiquement le diagramme d'impédances.

- Il s'agit de représenter dans le plan, la somme de trois vecteurs. En électricité, une pratique consiste à représenter à partir de l'origine, le vecteur $\overrightarrow{U_R}(zr, 0)$ puis à partir de son extrémité, le vecteur $\overrightarrow{U_L}\left(zl, \frac{\pi}{2}\right)$ et enfin également à partir de son extrémité, le vecteur $\overrightarrow{U_C} = \left(zc, -\frac{\pi}{2}\right)$.
- Importer dans votre script, les bibliothèques **TI PlotLib**. Modifier le script afin qu'il renvoie la valeur de l'impédance complexe après la représentation graphique. L'appui sur la touche del permettra de stopper l'affichage de la représentation graphique.



- Le vecteur correspondant à l'impédance totale sera en couleur magenta à l'aide de l'instruction **plt.color(255,0,255)**.

Unité 7 : Utiliser la bibliothèque cmath

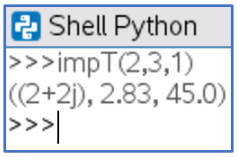
Application : Nombres complexes et sciences

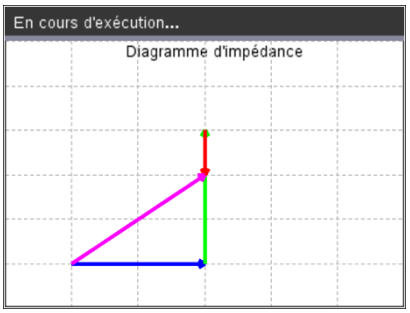
Dans cette application de l'unité 7, vous allez utiliser la bibliothèque **cmath** pour effectuer des calculs et représenter des nombres complexes utilisés en sciences physiques pour l'étude d'un circuit électrique.

Objectifs :

- Découvrir le module **cmath**.
- Utiliser les fonctions de la bibliothèque **cmath**.
- Représenter graphiquement des nombres complexes.
- Analyser un circuit électrique RLC série.

- Demander à nouveau l'exécution de votre script.
- Donner les arguments à la fonction permettant de calculer l'impédance totale **impT(2, 3, 1)**.
- Observer la représentation graphique du diagramme d'impédance.
- La touche del permet de retrouver le résultat du calcul précédent.





education.ti.france

@TIEducationFR

TIedtechFR

Contactez notre service client TI-Cares
education.ti.com/fr/csc
01 41 04 60 40

education.ti.com/fr